Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Analysis of the European airport network as a complex network

TITULACIÓ: Grau en Enginyeria d'Aeronavegació

AUTOR: Pol Gelabert Goday

DIRECTOR: Francesc Comellas Padró

DATA: 7 de setembre del 2018

Títol: Analysis of the European airport network as a complex network

Autor: Pol Gelabert Goday

Director: Francesc Comellas Padró

Data: 7 de setembre del 2018

Resum

La globalització juga un paper important en la societat a dia d'avui. Existeix la necessitat d'estar connectat i tenir la capacitat d'arribar a qualsevol lloc del món. Algunes de les xarxes o sistemes més importants que s'han creat són l'Internet, els sistemes de transport i comunicació, la xarxa elèctrica, les xarxes socials, etc., que poden ser catalogades com a xarxes complexes i cada una s'especialitza en una necessitat.

Per satisfer aquesta demanda, les xarxes han de tenir una alta resiliència enfront possibles errors en algun dels seus nodes, que poden succeir de manera aleatòria o per un atac intencionat. Davant una fallada d'un o múltiples nodes, la xarxa ha de ser capaç de redirigir la càrrega d'aquests als altres nodes de la xarxa, incrementant als nodes que resten operatius, la càrrega el més equitatiu possible.

Estudis recents han obtingut models sobre la caiguda en cascada dels nodes de xarxes reals com Twitter, Facebook o la xarxa elèctrica.

En aquest TFG estudiem el comportament d'una xarxa aeroportuària global, formada per aeroports d'Estats Units i d'Europa, davant d'una fallada en un conjunt d'aeroports. Concretament, es considera un radi d'actuació inicial a partir del qual es determinen els aeroports que fallen. Tenint present que cada aeroport, a més a més dels vols regulars, és capaç d'admetre una càrrega màxima, s'analitza com evoluciona la fallada en cascada per diverses càrregues. Es valora l'efectivitat de dues estratègies per forçar la fallada d'un aeroport i quines afectacions produeix a la xarxa. Per realitzar aquest estudi, primer hem analitzat les propietats de les xarxes aèries d'Estats Units, d'Europa i la xarxa global a partir de dades públiques. També interpretem l'evolució de les propietats de la xarxa d'Estats Units entre els anys 2001 i 2018.

Hem realitzat les simulacions i analitzat els resultats mitjançant el paquet "NetworkX" de Python, que permet la creació, manipulació i anàlisi de xarxes complexes.

Title: Analysis of the European airport network as a complex network

Author: Pol Gelabert Goday

Director: Francesc Comellas Padró

Date: September 7th 2018

Overview

Globalization plays an important role in modern society. It exist the necessity to be connected and able to reach every place. Some of the most important networks and systems include the Internet network, the transport and comunication system, the electric grid, social networks,.. These can be considered as complex networks and each of them has specialized in a necessity.

In order to satisfy the demand, networks must be resilient enough to fight against posible node failure, which can occur by a random or intentioned attack. When a node or set of nodes fail, the network has to reorganize its connections with other operative neighbor nodes and assign them the load of the initial node, being the most equitable possible.

Recent studies have created models about cascade failure of real networks as Twitter, Facebook or the electric grid.

In this final degree project we study the behaviour of the global airport network, composed by the United States and Europe networks when cascade failure exist. Specifically, an initial radius is set to determine the failing airports.Taking in account the fact that each airport can handle a maximum amount of flight transfers in spite of its scheduled flights, we have analysed the evolution of cadacade failure for different loads. Efffectivity of two failing methods is tested. Hence, properties of the United States, Europe and global networks are studied first, taking the data from public sources. Furthermore, an interpretation of the evolution of its propierties has been done, betwen years 2001 and 2018.

The simulations and analysis has been done by using the “NetworkX” Python package that makes it easier to create, handle and analise complex networks.

ÍNDEX

INDEX DE FIGURES

INDEX DE TAULES

1 INTRODUCCIÓ

Les xarxes són molt presents arreu. El món tecnològic que ens envolta constitueix un conjunt de xarxes. Les xarxes de comunicació, les xarxes elèctriques i la xarxa d'Internet tenen un gran impacte en el nostre dia a dia.

L'estudi de les xarxes des del camp de les matemàtiques es fa mitjançant la teoria de grafs. El punt de partida d'aquesta branca es remonta a l'any 1736 amb la publicació de la resolució del problema dels ponts de Königsberg pel matemàtic suís Leonhard Euler - *Solutio problematic as geometriam situs pertinentis* (*The Solution of a Problem Relating to the Theory of Position*) (Euler, 1736).

La ciutat pertanyent a Prússi Oriental, estava dividida pel riu Pregel en tres seccions i una illa anomenada Kneiphof. La qüestió d'aquest famós problema era intentar traçar la ruta pels set ponts de la ciutat de Königsberg sense repetir un pont dues vegades, també conegut com a camí eulerià.

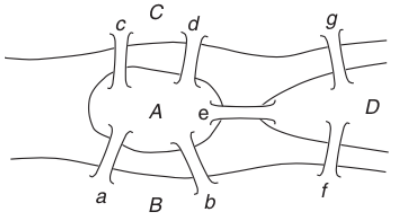

Figura 1 Ponts de Königsberg (1736)

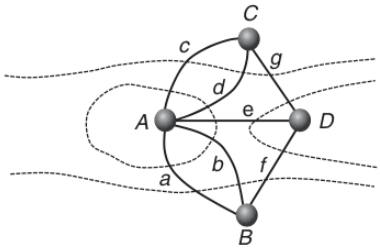

Figura 2 Graf representatiu del problema

El matemàtic es va fixar en l'estructura topològica de la xarxa, en la que cada secció és considerada com un punt (vèrtex) i els ponts mitjançant línies (enllaços). Les conclusions que va extreure són extrapolables a qualsevol altre xarxa d'aquest tipus: si més de dos vèrtex tenen grau senar, no existeix cap camí eulerià. Si dos vèrtex tenen grau senar i la resta el tenen parell, un camí eulerià pot existir si es comença des d'un dels vèrtexs amb grau senar i s'acaba en l'altre. Si tots els vèrtex tenen grau parell, és possible traçar un camí eulerià i, a més a més, constitueix un circuit eulerià (un camí que comença i acaba al mateix vèrtex).

Tot i ésser capaç de resoldre'l sense representar gràficament els ponts, ho va realitzar de tal forma que avui en dia seria l'equivalent al que anomenem graf. Aquest moment va esdevenir el naixement de l'anàlisi de les propietats matemàtiques de les xarxes. En els darrers anys s'han estudiat multitud de xarxes com la WWW, la xarxa elèctrica, les xarxes socials i la xarxa d'aeroports.

En aquest treball de fi de grau s'estudien les propietats de les xarxes d'aeroports d'Europa i d'Estats Units. A més, s'apliquen simulacions per estudiar el concepte de fallada en cascada dels aeroports d'aquestes xarxes.

1. Grafs

Un graf és una col·lecció de punts que poden estar units entre ells mitjançant línies. Aquests punts s'anomenen nodes o vèrtexs i constitueixen una entitat/objecte del sistema. Els nodes poden estar comunicats entre ells mitjançant un enllaç (aresta). Un graf es defineix com G, format per un conjunt de vèrtexs, V, un conjunt d'enllaços, E, i una funció φ, que associa els enllaços amb un parell de vèrtexs del graf que connecta:

$$G = (V, E, \varphi) \quad \textbf{(1.1)}$$

On,

$$V = \{v_1, v_2, v_3 \dots, v_m\}$$

$$E = \{e_1, e_2, e_3, \dots, e_m\}$$

Considerant que $v_i, v_i \in V$, aleshores $\varphi: e_p \rightarrow [v_i, v_j]$

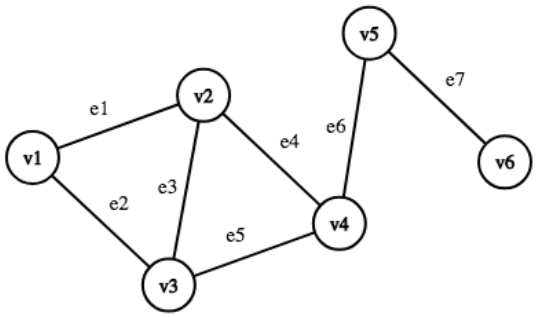

Figura 1.1 Exemple d'un graf

Els vèrtexs del graf i els enllaços les respectives unions. En aquesta figura, el conjunt de vèrtexs estan representats pels cercles i els enllaços per les unions entre els respectius nodes. L'expressió matemàtica del graf es la següent:

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

$$E = \{e_1, e_2, e_3, e_4, e_5, \square_6, e_7\}$$

$\varphi: e_1 \rightarrow [v_1, v_2]$ $\varphi: e_2 \rightarrow [v_1, v_3]$ $\varphi: e_3 \rightarrow [v_3, v_2]$

$\varphi: e_4 \rightarrow [v_2, v_4]$ $\varphi: e_5 \rightarrow [v_3, v_4]$ $\varphi: e_6 \rightarrow [v_4, v_5]$

$$\varphi: e_7 \rightarrow [v_5, v_6]$$

En nombroses ocasions, els nodes tenen associat uns atributs únics com per exemple el seu identificador, una descripció, les seves coordenades geogràfiques, etc. De la mateixa forma, els enllaços també poden tenir associat un atribut, generalment un nombre real anomenat pes $W(e)$. El graf estarà definit per $G = (V, W, \varphi)$ on $W = \{w_1, w_2, .., w_m\}$ és el conjunt dels pesos dels enllaços entre els nodes. L'ordre d'un graf es defineix com el nombre de vèrtexs que té, $n = |V|$ i la mida com el nombre d'enllaços entre nodes, $m = |E|$. La mida d'aquest graf és 6, mentre que l'ordre val 7.

Es poden diferenciar les diverses terminologies d'un graf:

- Graf nul: graf que no presenta cap vèrtex.
- Graf buit: graf que no conté cap enllaç.
- Graf trivial: graf que només té un node.
- Enllaços paral·lels: dos o més enllaços en que connecten els mateixos nodes.
- Enllaç bucle: enllaç el qual inicia i acaba al mateix vèrtex.
- Enllaços adjacents: parell d'enllaços que comparteixen un mateix vèrtex.
- Vèrtexs adjacents: vèrtex connectats per un enllaç $[v1, v2]$.
- Graf connex: graf en que cada parell de nodes està connectat per un enllaç. Informalment, des d'un node qualsevol es pot arribar a tots els altres nodes del graf mitjançant els successius enllaços d'aquest.
- Graf simple: aquell que no conté cap bucle ni enllaç paral·lel. La figura 1. N'és un exemple.
- Multigraf: graf que permet enllaços paral·lels i enllaços bucle.
- Digraf o graf dirigit: graf en que els enllaços són unidireccionals: $[v1, v2]$ es considera que v_1 enllaça amb v_2 però no a l'inrevés.

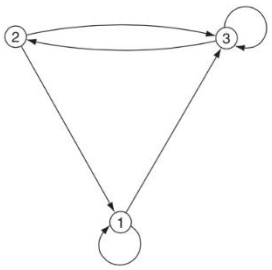

Figura 1.2 Exemple de multigraf, amb pesos i enllaços bucle.

En la figura 1.2, la funció φ pren els següents valors:

$\varphi: e_1 \rightarrow [v_1, v_1]$ $\varphi: e_2 \rightarrow [v_2, v_1]$ $\varphi: e_3 \rightarrow [v_2, v_3]$

$\varphi: e_4 \rightarrow [v_3, v_2]$ $\varphi: e_5 \rightarrow [v_3, v_3]$ $\varphi: e_6 \rightarrow [v_1, v_3]$

S'associa al graf G la matriu d'adjacència, $A(G)$. Aquesta matriu representa els enllaços entre vèrtex. Si existeix l'enllaç $[v_i, v_j]$, la matriu A conté un 1 a la posició A_{ij} i a la A_{ji}. Prenent com a G el graf representat en la figura 1.1, la següent matriu exemplifica la matriu d'adjacència del graf G:

$$A(G) = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad \textbf{(1.2)}$$

S'anomena camí aquell recorregut que comença al vèrtex u i acaba al vèrtex v sense passar dues vegades pel mateix enllaç. Es defineixen dos tipus de camins:

- Camí hamiltonià: Camí el qual es recorren tots els vèrtex del graf inclosos i sense repetir-los (no cal recórrer tots els enllaços).
- Camí eulerià: tenint en compte que es poden repetir els nodes, constitueix un camí eulerià si es recorren tots els enllaços del graf acabant al mateix node d'origen.

2 Xarxes complexes

Les xarxes complexes són molt presents en la nostra vida quotidiana, des del món tecnològic i les xarxes socials, fins a les xarxes biològiques. Conèixer-les amb el major detall possible permet crear models que reprodueixen des del seu comportament intern com a xarxa fins a les interaccions amb altres xarxes complexes. La finalitat d'aquests estudis són millorar les propietats de la xarxa per, globalment, millorar-ne la seva eficiència.

2.1 Tipus de xarxes

2.1.1 Xarxes biològiques

Un tipus de xarxes que formen part de la naturalesa i l'anatomia dels éssers vius són les xarxes biològiques. El seu estudi, anàlisi i modelatge són tasques importants en la vida tecnològica. La majoria d'aquestes xarxes encara romanen lluny de la realitat de la naturalesa degut a la interpretació de la complexitat de les relacions i les peculiaritats de les dades. *Gene coexpression network* o també anomenada *Weighted correlation network analysis (WCNA)*, és una xarxa complexa unidireccional amb pes. Aquesta xarxa captura la informació de la interacció dels gens en diverses condicions biològiques, com per exemple durant la divisió cel·lular, o quan les cèl·lules reaccionen en un determinat tractament de medicaments. La xarxa WCNA té l'expressió $G = (V, E, \varphi)$, on el conjunt de nodes V representen els gens, el conjunt d'enllaços entre els gens, E, representen les co-expressions en les diferents condicions, i els pesos φ dels enllaços relacionen el parell de gens que interaccionen/s'activen sota les mateixes condicions biològiques. Un altre exemple n'és la xarxa proteïna-proteïna, que relaciona les interaccions entre aquestes en funció del tipus d'interacció que es tracti. També constitueix un graf unidireccional en què els nodes representen el conjunt de proteïnes, els enllaços representen els enllaços entre les proteïnes i φ les interaccions entre les proteïnes, com per crear una nova proteïna complexa o per activar-la.

2.1.2 Xarxes socials

Lluny de l'àmbit tecnològic, un altre tipus de xarxes d'interès científic són les xarxes socials. Un xarxa social és, normalment, una xarxa de persones, encara que poden ser una xarxa de grups de persones, com les empreses. Les persones o grups formen els nodes, mentre que els enllaços representen algun tipus de connexió entre ells, com l'amistat entre dos individus o la relació professional entre empreses. L'anàlisi de xarxes socials (SNA) és el procés d'estudi de l'estructura d'aquestes. *Facebook*, n'és la xarxa social del món occidental per excel·lència amb més de 2 100 milions de nodes.

2.1.3 Xarxes blockchain

Blockchain o cadena de blocs és una estructura de dades en que la informació continguda s'agrupa en blocs relacionats entre ells mitjançant mètodes criptogràfics. Va ésser inventat anònimament sota el pseudònim Satoshi Nakamoto al 2008 amb la finalitat d'usar-se com a llibre de comptabilitat (*Ledger*) de la criptomoneda *Bitcoin*. Un dels beneficis més rellevants de les criptomonedes és la naturalesa pública del *ledger*, ja que és globalment compartit.
Bitcoin és considerada la criptomoneda mare dins el món blockchain. Va començar com un projecte i ha evolucionat fins al punt de permetre realitzar transaccions monetàries al món real. Basat en la criptografia de clau pública i privada (asimètrica), un usuari (emissor) que envia Bitcoins a un altre (receptor), es xifra la transacció amb la clau púlica del receptor i aquest desxifra el missatge usant la seva clau privada. La seguretat d'aquesta metodologia recau en les funcions matemàtiques unidireccionals anomenades *Hash*, que són les que generen les claus privades a partir de les públiques. Un cop generada la clau privada, és quasi bé impossible de conèixer-la per als altres usuaris, fet que converteix aquesta la criptografia asimètrica en la més segura actualment. Un cop realitzada la transacció i verificada per la majoria dels usuaris de la xarxa, es guarda al blockchain, una llista cronològica pública que conté totes les transaccions de la xarxa Bitcoin. Transcorreguts deu minuts, es tanca aquest bloc i es genera un de nou. Gràcies a aquest sistema descentralitzat basat en la transparència, ens permet estudiar el sistemes financers que prèviament eren impossible el seu estudi degut a la privacitat de les transaccions.
Tenint accés a un determinat bloc, es poden extreure totes les transaccions i reconstruir el bloc de transaccions de la criptomoneda. Aquest graf proporciona informació sobre la tendència dels bitcoins entre les adreces públiques al llarg del temps. Concretament, és un graf dirigit en que els nodes denoten les adreces públiques dels usuaris anònims i cada enllaç dirigit representa la transacció en particular d'una adreça origen a una altra destí.

2.1.4 Xarxes tecnològiques

Els sistemes de transport, junt amb altres infraestructures com la xarxa elèctrica o de comunicació, són elements fonamentals en la nostra societat i economia. Garanteixen un alt nivell de mobilitat que tothom experimenta, i que és vital per la cohesió dels mercats i la qualitat de vida dels ciutadans. A més a més, els sistemes de transport permeten un creixement socio-econòmic i nombrosos llocs de treball. També cal destacar, per altra banda, la inestabilitat d'aquestes infraestructures enfront errors aleatoris o intencionats atacs terroristes, en què la totalitat de la societat és afectada.

La figura 2.1 mostra al xarxa de vols d'una aerolínea d'Estats Units, on els aeroports estan connectats si tenen una ruta directa. És observable que els

aeroports de Houston, Nova York i Cleveland actuen com a *hubs* d'aquesta aerolínea, oferint nombroses rutes cap altres possibles destinacions.

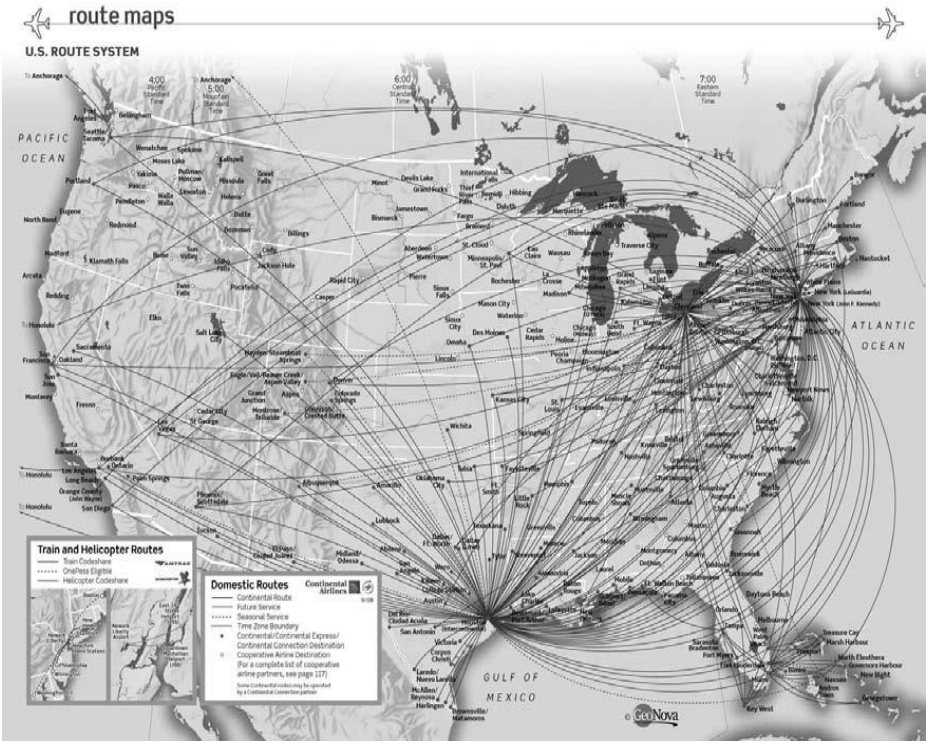

Figura 2.1 Xarxa d'interconnexions entre aeroports d'Estats Units [link2bibl]

2.1.5 Xarxa de computadors

Les xarxes de computadors constitueixen les xarxes idònies per estudiar el seu rol dins el món de l'Internet. Enllaçats físicament, tot i que poden estar connectats per satèl·lit, formen des de xarxes LAN (*Local Area Network*) fins a xarxes WAN (*Wide Area Network*). Rarament, les xarxes de computadors d'avui dia estan aïllades; quasi bé totes elles estan connectades a la xarxa de xarxes, Internet. Internet està està format per nombrosos sistemes autònoms (AS) que cooperen intercanviant petits paquets d'informació entre ells. Aquests paquets són transferits als *routers "designated computers",* contenint totes les taules que permeten encaminar correctament la informació fins al destinatari. L'Internet, com a xarxa complexa es pot estudiar al nivell de *router* – els *routers* esdevenen nodes enllaçats físicament- i al nivell de sistema autònom – cada AS és un node interconnectat amb un altre AS. L'any 2009, la xarxa de sistemes autònoms constava de 10^4 nodes, tot i que existien centenars de milers més de forma aïllada.

2.1.6 Xarxa WWW

Un tipus de xarxa més abstracta és la *World Wide Web* (WWW). Sovint s'usen indistintament els termes Web i Internet, però tècnicament són molt diferents. L'Internet és una xarxa física d'ordinadors enllaçats per cable o a vegades radio enllaços entre ells. En canvi, la Web és una xarxa d'informació emmagatzemada en documents d'hipertext (pàgines web). La WWW constitueix una xarxa formada per bilions de pàgines web escrites en llenguatge HTML (*HyperText Markup Language*). Els vèrtex de la WWW són les pàgines web i els enllaços els hipervincles; els textos subratllats o botons que al prémer, el navegador carrega l'URL (*Uniform Resource Locator*) especificada en l'hipervincle, fet que condueix d'una pàgina web a una altra.

2.2 Propietats principals

Amb la finalitat d'estudiar els sistemes complexes, és necessari conèixer les seves propietats fonamentals que componen la seva estructura. Es defineix el següent graf, anomenat *G*, a partir del qual es detallaran les principals propietats.

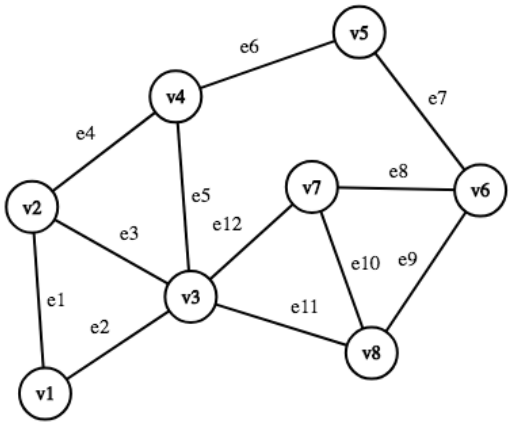

Figura 2.2 Graf format per vuit nodes i onze enllaços.

2.2.1 Distància

La distància, $d(i,j)$ és el nombre d'enllaços requerits traçant el camí més curt entre el node i el node j d'un graf. Si es tracta d'un graf amb pes, el camí òptim serà aquell en què el sumatori de pesos sigui el menor.

Es determina la matriu de les distàncies del graf a partir de la distància més curta entre els nodes:

$$d(i,j) = \begin{bmatrix} 0 & 1 & 1 & 2 & 3 & 3 & 2 & 2 \\ 1 & 0 & 1 & 1 & 2 & 3 & 2 & 2 \\ 1 & 1 & 0 & 1 & 2 & 2 & 1 & 1 \\ 2 & 1 & 1 & 0 & 1 & 2 & 2 & 2 \\ 3 & 2 & 2 & 1 & 0 & 1 & 2 & 2 \\ 3 & 3 & 2 & 2 & 1 & 0 & 1 & 1 \\ 2 & 2 & 1 & 2 & 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 4 & 2 & 1 & 1 & 0 \end{bmatrix} \quad \textbf{(2.2)}$$

Per tal de determinar la distància mitjana del graf, és necessari calcular la mitjana de la distància únicament per a les parelles de nodes que estan connectats:

$$\bar{d} = \frac{1}{n(n-1)} \sum_{j \in V(G)} d(i,j) \quad \textbf{(2.3)}$$

2.2.2 Excentricitat

Es defineix l'excentricitat d'un determinat node, $e(i)$, com la màxima distància entre aquest i qualsevol altre node del graf, x.

$$e(i) = max_{x \in V(G)}\{d(i,x)\} \quad \textbf{(2.4)}$$

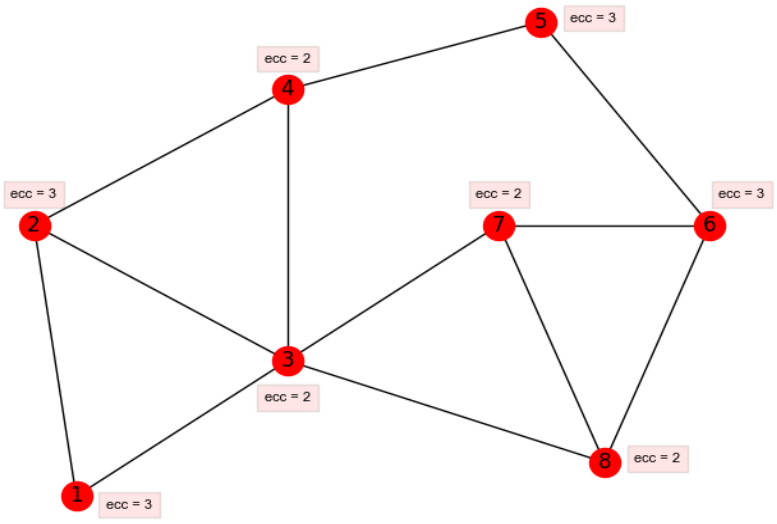

Figura 2.3 Excentricitat del graf analitzat de la figura 2.2

2.2.3 Diàmetre i radi

El diàmetre d'un graf és la màxima excentricitat de qualsevol node d'aquest, és a dir, la distància màxima entre dos nodes del graf.

$$D = max_{i,j \in V(G)}\, d(i,j) \tag{2.5}$$

El valor del radi és la mínima excentricitat dels nodes, i el node que presenta una excentricitat igual al radi s'anomena node central. Fixant-nos en el graf *G*, el diàmetre és 3 i el radi val 2.

2.2.4 Coeficient d'agrupament o *clustering*

En teoria de grafs, el coeficient d'agrupament o *clustering* mesura el grau de connectivitat local d'un graf, és a dir, l'agrupament que existeix entre els nodes d'aquest (si els veïns d'un node són també veïns entre si).

Per calcular aquest coeficient cal fixar-se en el grau del node i, i el seu nombre d'enllaços.
Si N_i és el nombre de branques que uneixen els nodes adjacents a i respectivament, k_i el grau del vèrtex i, es pot establir el coeficient d'agrupament d'un determinat vèrtex com:

$$C_i = \frac{2N_i}{k_i(k_i-1)} \tag{2.6}$$

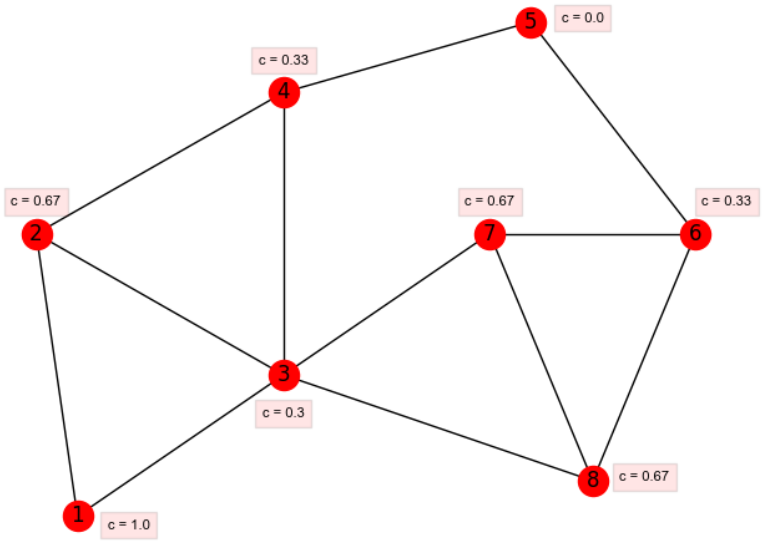

Figura 2.4 Coeficient d'agrupametn dels nodes del graf G.

Aleshores, el coeficient d'agrupament global de G, denotat C_G, és la mitjana de C_i sobre tots es vèrtexs $i \in V(G)$:

$$C_G = \frac{1}{n}\sum_{i=1}^{n} C_i \quad \textbf{(2.7)}$$

El coeficient d'agrupament sempre està comprès entre el rang $0 \leq C \leq 1$. A la figura 2.4 es pot observar el coeficient d'agrupament de cada node. Un resultat del coeficient d'agrupament igual a 1 implica que tots els nodes adjacents al node d'estudi estan connectats entre ells. El vèrtex 1 té $C = 1$ ja que els seus dos nodes adjacents (2,3) estan enllaçats. Un coeficient d'agrupament igual a 0 significa que cap dels nodes adjacents estan connectats entre ells.

2.2.5 Distribució de graus

El grau d'un node és igual al nombre de nodes adjacents a aquest. En grafs dirigits, es pot diferenciar entre els enllaços entrants i els enllaços sortints, així com el grau total obtingut a partir de la suma d'ambdós.
La distribució de graus en un graf pot caracteritzar-se per una funció de distribució P(k), que dóna la probabilitat que un node triat a l'atzar tingui grau k. Així, la representació gràfica de la distribució de graus és la fracció de vèrtexs de la xarxa que tenen grau k :

$$p(k) = {}^{n(k)}/_{n} \quad \textbf{(2.8)}$$

En una xarxa aleatòria la seqüència de graus segueix una distribució de Poisson:

$$P(k) = \frac{\lambda^k \cdot e^{-\lambda}}{k!} \quad \textbf{(2.9)}$$

El gràfic d'aquesta distribució presenta un pic a $\bar{k}$, tal i com es pot veure a la figura 2.5 a). La probabilitat de trobar un vèrtex amb k branques és negligible per a $k >> \bar{k}$ i $k << \bar{k}$.

Moltes xarxes reals de grans dimensions tenen una distribució de graus diferent a la distribució de Poisson i segueixen, en molts casos, una llei potencial on $P(k) \sim k^{-\lambda}$. Aquest tipus de xarxes es diu que presenten invariància d'escala (*scale-free*) ja que al afegir més nodes a la xarxa no canvia el factor d'escala de la seva distribució dels graus. Per exemple, la xarxa WWW segueix una distribució de llei potencial pel fet que ha mantingut la seva distribució de graus tot i haver augmentat considerablement el nombre de nodes i enllaços en els últims anys.

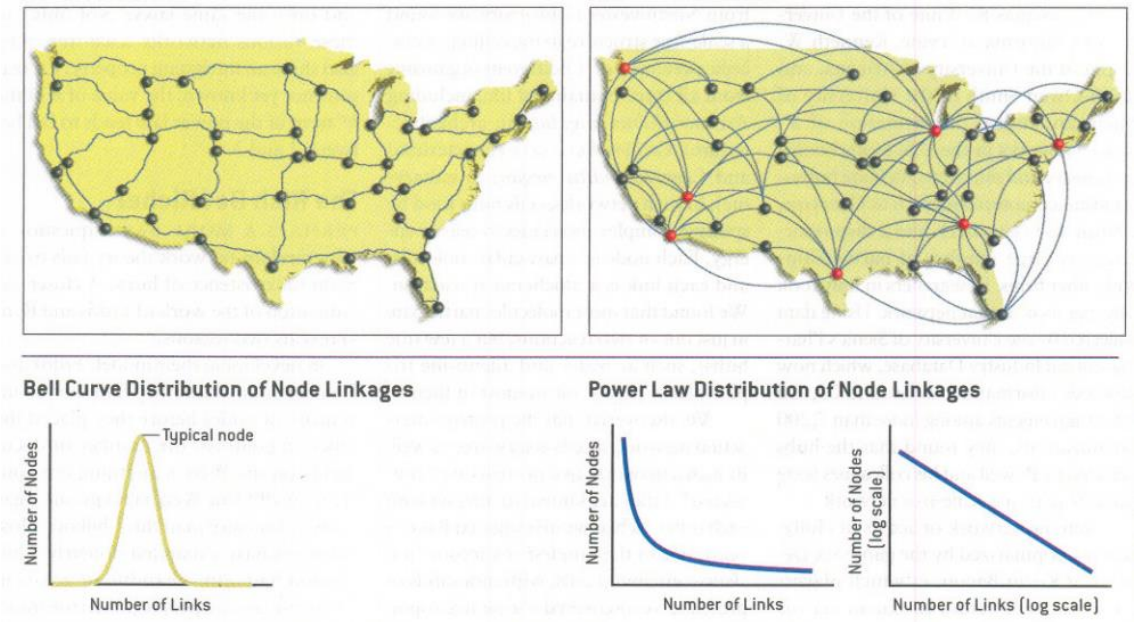

Figura 2.5 a) Xarxa de transport terrestre dels Estats Units amb distribució de graus de Poisson. b) Xarxa de transport aeri amb distribució de llei potencial.

Les xarxes aleatòries, que s'assemblen al sistema de carreteres d'Estats Units (2.5 a) estan formades per nodes amb enllaços distribuïts aleatòriament. Aquesta xarxa exemplifica la distribució graus de Poisson en què la majoria de nodes tenen els mateixos enllaços connectant amb altres carreteres. Per altra banda, la figura 2.55 b), mostra la distribució de graus de les rutes d'una aerolínia nordamericana, que segueix una llei potencial. Els aeroports que actuen com a *hub* (vermell) presenten un gran nombre d'enllaços, però representen una petita fracció del total de nodes de la xarxa, que, tenen poques connexions a altres aeroports. Un tret característic de les xarxes que amb distribució de llei potencial, és que al representar-los en escala logarítmica, resulta ser una línia de pendent constant.

2.2.6 Correlació entre graus

Una altra característica interessant de les xarxes complexes és l'existència de diferents tipus de correlació entre graus. La correlació entre graus mesura de quina manera estan connectats els nodes d'un determinat grau. És important tenir present que en una xarxa aleatòria s'assumeix que no hi ha correlació entre graus de nodes veïns, però en xarxes reals sí s'estudia aquest paràmetre. Tenint en compte que una correlació positiva o *assortative* és aquella en què els nodes d'alt grau tendeixen a estar connectats entre ells. Contràriament, una correlació negativa o *dissortative* es aquella en què els nodes d'alt grau tendeixen a estar enllaçats amb nodes de grau baix.

Una forma fàcil de quantificar si una xarxa és *assortative* o no, és calculant el coeficient de correlació dels graus dels nodes existents a la xarxa. Newman [13] va proposar la definició d'un coeficient per explicar aquest comportament. La distribució del grau excedent $q(k_j)$, essent $k+1$ el grau total, ve donada pel grau del node, k_i, i el grau del node al qual està connectat, k_j, així com per la probabilitat $p(k_j)$ que un node escollit a l'atzar tingui grau k_j.

$$q(k_j) = \frac{(k_j+1)p(k_j+1)}{\sum_i k_i p(k_i)} \quad \textbf{(2.10)}$$

Definint el coeficient d'aparellament com:

$$r = \frac{\sum_{k_i \square_j} k_i k_j [e(k_i k_j) - q(k_i)q(k_j)]}{\sigma_q^2} \quad \textbf{(2.11)}$$

On σ_q és la desviació estàndard de la distribució $q(k_j)$. Ara bé, aquest no es pot definir per a xarxes regulars[3].
En cas que es tracti de xarxes dirigides aquest *coeficient és el coeficient de correlació de Pearson* dels graus dels nodes en ambdós extrems de l'enllaç.

$$r = \frac{m^{-1}\sum_e k_i(e)k_j(e) - [m^{-1}\sum_e \frac{1}{2}[k_i(e)+k_j(e)]]^2}{m^{-1}\sum_e \frac{1}{2}[k_i^2(e)+k_j^2(e)] - [m^{-1}\sum_e \frac{1}{2}[k_i(e)+k_j(e)]]^2} \quad \textbf{(2.12)}$$

On e és l'enllaç tal que el grau dels seus extrems són $k_i(e)$ i $[k_j(e)$, i m és la mida del graf.

2.2.7 Densitat de connexió

La densitat de connexió es la proporció del nombre actual d'enllaços de la xarxa i el nombre total d'enllaços possibles que la xarxa pot tenir. La xarxa té un límit de connexions, ja que com definit en el capítol 1, si presentés més del total, deixaria de ser un graf simple i seria un multigraf. És l'estimador més simple del cost físic d'una xarxa. Una densitat de connexió alta incrementa el coeficient d'agrupament del graf, mentre que connexions entre nodes distants o entre clústers diferents disminueixen aquest paràmetre.

Exemples de densitat de connexió altes són: l'expansió d'una malaltia contagiosa, la xarxa neurològica del cervell o la xarxa de telecomunicacions.
El paràmetre de densitat de connexió (D) es calcula:

[3] Xarxes regulars: Tots els vèrtex de la xarxa tenen el mateix grau.

$$D = \frac{2m}{N(N-1)} \quad \textbf{(2.13)}$$

On m és la mida del graf i N l'odre d'aquest, segons les definicions del primer capítol.

2.2.8 Index d'Estrada

L'índex d'Estrada [2] d'un graf G es defineix com $EE(G) = \sum_{i=1}^{n} e^{\lambda_i}$, on $\lambda_1, \lambda_2, \dots \lambda_n$ són els valors propis de G. Aquest índex presenta els següents límits inferior i superior:

$$n \ll EE(G) \ll e^{n-1} + \frac{n-1}{e} \quad \textbf{(2.14)}$$

2.2.9 Index de Wiener

L'índex de Wiener [2] estima la densitat de l'estructura del graf. És un dels índex topològics que van tenir força importància durant l'estudi de la part matemàtica de la química. El motiu principal és que es poden calcular a partir de grafs moleculars i correlar-se amb les propietats, tant físiques com químiques, de la molècula corresponent. Ha estat emprat per obtenir diversos objectius: la correlació amb diferents paràmetres termodinàmics (Harry Wiener, 1947), predir constants crítiques (Stiel i Thordos, 1962), la correlació amb la densitat, viscositat i tensió superficial (Rouvay i Craffor, 1976), entre d'altres. Aquest índex fa referència al temps total que un node Es defineix com la suma del les distàncies dels camins més curts, $d(i,j)$, entre tots els parells de nodes de la xarxa:

$$W(G) = \frac{1}{2}\sum_i \sum_j d(i,j) \quad \textbf{(2.15)}$$

Aquest índex està acotat pels següents límits:

$$\frac{n(n-1)}{2} \ll W(G) \ll \frac{n(n^2-1)}{6} \quad \textbf{(2.16)}$$

En funció de la proximitat de $W(G)$ al límit inferior denota que la xarxa és poc densa, mentre que si és proper al límit superior indica una alta densitat en la xarxa.

2.2.10 Index Q

Ernesto Estrada [1] proposa aquest índex per a determinar si una xarxa està eficientment configurada.

Prèviament, es la distància de comunicació, $\xi(\beta)$, en funció de la comunicabilitat entre dos nodes p i q:

$$G_{pq} = \left(\sum_{k=0}^{\infty} \frac{A^k}{k!}\right) \quad \textbf{(2.17)}$$

$$\xi_{p,q}(\beta) = G_{pp}(\beta) + G_{qq}(\beta) - 2G_{pq}(\beta) \quad \textbf{(2.18)}$$

On el paràmetre β és la temperatura inversa definida:l'invers de la temperatura per la constant de Bolzano:

$$\beta = \frac{1}{kT} \quad \textbf{(2.19)}$$

L'índex de la distancia de comunicabilitat, estarà definit per:

$$\gamma(G) = \frac{1}{2}\sum_{p,q} \xi_{p,q} \quad \textbf{(2.20)}$$

Tornant al Índex Q, la seva fórmula és la següent:

$$Q = \ln\left(\frac{0.8578W}{\gamma}\right) \quad \textbf{(2.21)}$$

On la constant 0.8578 correspon a la distància de comunicabilitat entre un parell de nodes de K_n amb $\beta^1 = 1$, W és l'índex de Wiener i γ és l'índex de la distància de comunicació.

Per valors de $Q > 0$, indica que la transmissió usant el camí més curt triga més temps que usant la distancia de comunicabilitat, implicant que $\frac{0.8578W}{\gamma} < 1$. En altres paraules, el graf G no és eficient en termes de comunicació entre els nodes i s'etiqueta com a graf flexible. La majoria d'aquests tipus de grafs són aquells en què els enllaços estan distribuïts en un pla de dues dimensions, com per exemple la xarxa de carrers d'una ciutat o un circuit electrònic.

Contràriament, valors de $Q < 0$ indiquen que les connexions són eficientment ja que el camí més curt triga menys que la distància de comunicabiliat. Es denominen grafs rígids. Estan geogràficament limitats a un espai determinat, que pot ser una territori geogràfic, o una regió d'un organisme biològic com la xarxa cerebral i la xarxa de neurones. Tots ells, utilitzen eficientment l'espai; partint sempre d'una regió plana, l'enllaç utilitza l'espai tridimensional o encara més espais multidimensionals per incrementar la connectivitat de la xarxa. Per exemple, l'índex Q de la xarxa de transport aeri de l'any 1997 és -16.58 .

3 Estudi de xarxes aeroportuàries

Nombroses xarxes complexes poden ésser representades de diverses formes. Per exemple, la xarxa d'Internet es pot representar com un conjunt de nodes (les pàgines web) connectades mitjançant enllaços (hiperlinks). El mateix graf també es pot representar considerant els *routers* com nodes i les seves connexions físiques com els enllaços.
Aquesta dualitat en la representació d'una xarxa és compartida també per la xarxa de transport aeri. El sistema de transport aeri està compost per molts elements de diferent tipologia, que interaccionen i treballen conjuntament.
Una possible aproximació és analitzar la mobilitat dels passatgers, establint com a focus d'estudi el flux de passatgers entre les principals ciutats i països del món. Aquesta recerca és de gran importància des d'un punt de vista social. No és sorprenent que la majoria d'aquests estudis s'han centrat en la mobilitat dels passatgers, desestimant altres aspectes tècnics importants. És per aquest motiu, que la construcció d'una xarxa del sistema aeroportuari i posterior anàlisi és inevitable. Els nodes representen els aeroports i un enllaç entre ells és creat quan existeix un vol directe que els connecta. La naturalesa de la xarxa d'aeroports constitueix un graf dirigit, en què un enllaç entre el node A i B, descriu una connexió d'origen A i destí B, però no a l'inrevés. Cal tenir en compte que la majoria de vols entre dos nodes són recíprocs, ja que al existir un vol entre A i B, aquest mateix avió realitzarà un vol de tornada a A en una elevada probabilitat. Aquest fet no es compleix en vols que realitzen escales tècniques o comercials, en situacions en què el trajecte d'anada i tornada tenen associats diferents rutes ni en vols que realitzen una parada a l'aeroport alternatiu degut a una emergència a bord. Tot i aquestes situacions, minoritàries respecte la majoria de vols, es consideren negligibles. És per aquest motiu que es considera que el graf que representa la xarxa aeroportuària és un graf simple, assumint que tota connexió entre A i B és recíproca.

No obstant, al graf de la xarxa aeroportuària s'ha assignat un pes a cada ruta. Aquest pes, que segueix la funció W, representa el nombre anual de vols entre un node A i B. Així doncs, la definició del graf és $G = (V, W, \varphi)$, tal i com està especificat en el primer capítol.

En aquest estudi s'analitzen les propietats de les xarxes d'aeroports d'Europa, d'Estats Units i una xarxa global, suma de les anteriors dues xarxes. El rang de les dates que compren l'estudi és des de l'1 de maig del 2017 fins el 31 d'abril de 2018.

Addicionalment no tots els vols són equivalents. El nombre de places disponibles en cada avió és diferent, oscil·la entre 50 seients d'un jet regional petit, fins els 853 seients d'un Airbus A380.

3.1 Formació de grafs

En aquesta secció s'explica detalladament com s'han obtingut les dades i el posterior processament per generar els tres grafs a partir dels quals es centra l'estudi.

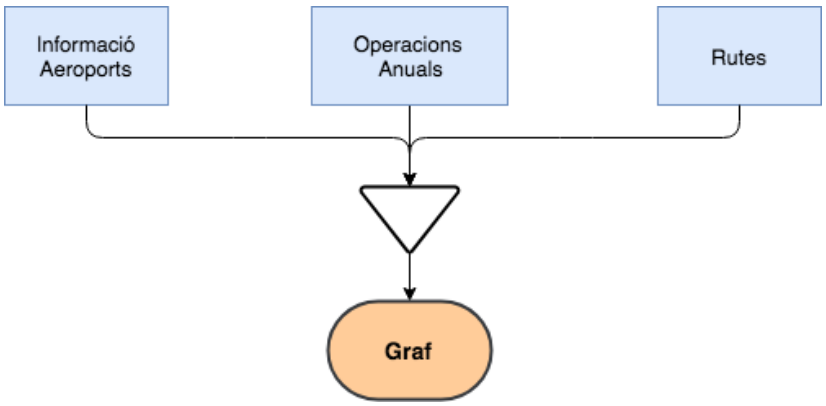

Figura 3.1 Diagama de flux del procés d'adquisició de les dades.

La figura 3.1 mostra el diagrama de flux per tal de generar un graf d'una xarxa aeroportuària de forma genèrica. La metodologia usada per les tres fonts d'informació ha estat descarregar els arxius en format *.csv*, convertir-los a format *.xlsx* i posteriorment analitzar-los mitjançant l'eina *Enthouht Canopy*, usant el llenguatge *python*.

3.1.1 Obtenció de dades

El primer pas per generar els tres grafs és obtenir la informació bàsica dels aeroports d'estudi. Aquesta informació s'ha extret del web Datahub.io i engloba:

- Codi ICAO
- Codi IATA
- Tipus d'aeroport
- Nom de l'aeroport
- Continent
- País
- Regió
- Municipi
- Latitud
- Longitud

El nombre de vols anuals de cada aeroport tant d'Eurocontrol com d'Estats Units és necessari realitzar-lo de la següent forma:

- **Eurocontrol:** les operacions anuals de la xarxa aeroportuària d'Eurocontrol s'han extret de la plataforma *Statistics and forecasts (STATFOR)* del web d'Eurocontrol.
  Més concretament, s'ha generat un arxiu per cada mes del període estudiat, (i), que conté la mitjana de vols $(\bar{X}_i)$ diaris de cada aeroport. El nombre total de vols anuals (aF) s'obté:

$$aF = \sum_{i=1}^{12} \bar{X}_i \cdot k_i \qquad \textbf{(3.1)}$$

  On k és el nombre de dies del mes i. aF no té en compte entre quins nodes s'han realitzat els vols, únicament és d'interès perquè prové de la institució reguladora del trànsit aeri del continent europeu. És per aquest motiu, la necessitat obtenir d'una tercera font, les rutes en les quals s'han realitzat aquests vols.

- **Estats Units:** la plataforma ATADS (Air Traffic Activity System) [9] de l'Administració Federal de l'Aviació d'Estats Units (FAA) permet a qualsevol usuari web consultar i extreure dades sobre els seus aeroports. Usant els diferents filtres com les dates d'estudi, els aeroports d'interès i certes opcions sobre quins vols mostrar (Itinerants, locals i VFR) s'obté l'informe en format .xls.

Per al tercer i últim procés previ a generar els grafs, s'obtenen les rutes entre els aeroports estudiats. Opensky-Network [10] és una comunitat dedicada a l'ús de receptors per crear una xarxa en la que es monitoritzi contínuament i en temps real, la informació de la vigilància del tràfic aeri (missatges ADS-B (*Automatic Dependent Surveillance-Broadcast*) i missatges mode S). Compta amb més de 1030 sensors arreu del món, i ja ha captat més de 9 trilions de missatges des de l'any 2016, l'equivalent a monitoritzar 300.000 vols diaris. La figura 3.2 mostra les zones del mapamundi cobertes pels receptors. Europa està coberta en la seva totalitat i Estats Units ho està en més d'un 90%.

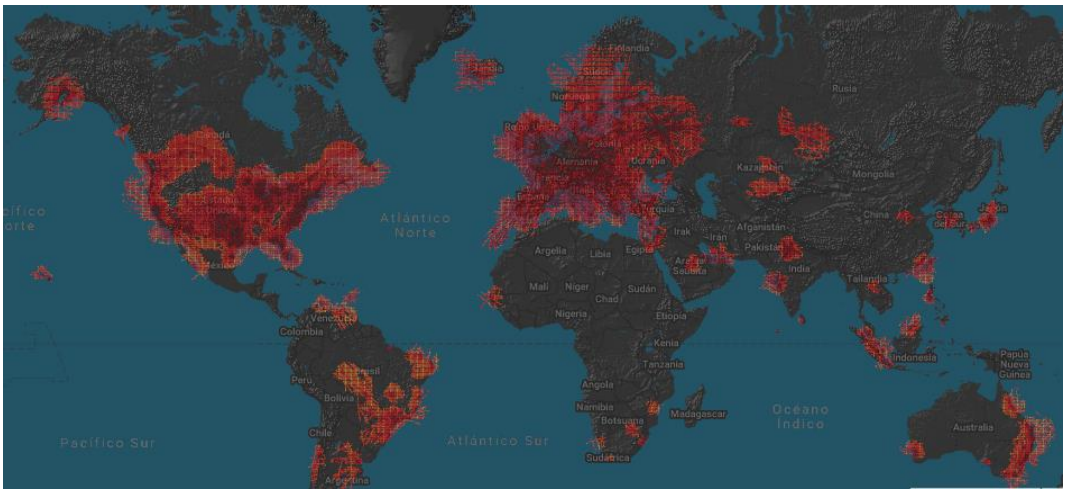

Figura 3.2 Mapamundi de la cobertura radar d'Opensky Network

Un altre dels punts forts d'aquesta comunitat és l'accés per a recercadors registrats a la base de dades SQL on es registra tota la informació històrica dels vols. Es pot accedir a través d'un client SSH. En aquest estudi s'ha usat *PuTTY* per tal d'enregistrar el resultat de la consulta en un fitxer de text. La consulta és la següent:

> *select estdepartureairport, estarrivalairport, count(*) Count_Duplicate from flights_data4 where day>=1493596801 and day<=1525132799 and estdepartureairport is not null and estarrivalairport is not null group by estdepartureairport, estarrivalairport having count(*) >= 1 order by count(*) desc;quit;*

Analitzant-la detalladament, mostra el recompte de registres duplicats entre tots els aeroports durant l'any d'estudi, que justament són els vols anuals de cada ruta entre un node A i un B. El període d'interès cal escriure'l en format *timestamp* i els registres que no tenen o origen o destí, els omet.

Tot i la gran quantitat de missatges rebuts, hi ha un petit percentatge de dades que no es registren o bé perquè en l'origen o el destí tenen camps vuits, per limitacions dels receptors que no capten senyals MLAT (*Multilateració*) o els minoritaris avions amb certa antiguitat sense ADS-B. Segons un estudi realitzat per diverses universitats d'Alemanya, Suïssa i Anglaterra l'any 2016, el 70% dels vols amb transponedors Mode S, estan equipats de l'ADS-B a bord. No obstant, la suma de tot el recompte de vols obtinguts per a cada aeroport comparat amb el valor esperat d'Eurocontrol, en determinats casos s'observen aquestes mancances.

Per aquest motiu, s'ha realitzat de forma individual per cada aeroport un ajust del nombre de vols. Específicament, en aquells aeroports que, degut a la l'orografia del terreny s'han monitoritzat menys missatges o perquè són aeroports on no hi han masses receptors a la vora, es realitza el següent ràtio (r_i):

$$r_i = \frac{\sum_{e=1}^{e=\max(\#veins\ de\ i)} w_i}{\#Vols\_Eurocontrol_i} \qquad \textbf{(3.2)}$$

On el numerador representa el sumatori dels pesos (w_i) per a tots els enllaços, (e), d'un aeroport i. Un cop calculat el ràtio, s'aplica a cada enllaç de cada node:

$$nou\_w_{i,e} = r_i \cdot w_i \qquad \textbf{(3.3)}$$

3.1.2 Graf d'Eurocontrol

El primer graf a generar és el que compren la xarxa d'aeroports d'Eurocontrol. És important mencionar que aquest graf a més a més d'incloure els estats membres d'aquesta organització, incorpora també República de l'Azerbaidjan i Islàndia.

Un cop s'han enregistrat els fitxers necessaris, es genera el graf d'Eurocontrol $G\_E = (V, W, \varphi)$, segons l'especificació del capítol 1.

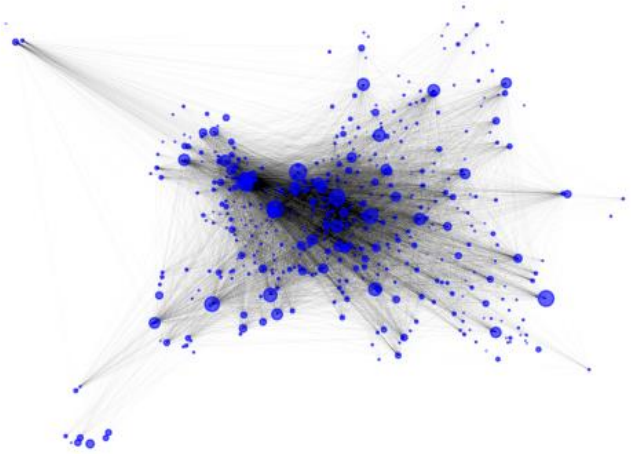

Figura 3.3 Graf de la xarxa d'Eurocontrol.

3.1.3 Graf d'Estats Units

El segon graf corresponent al segon país més extens del continent Americà és relativament menys complex que el d'Eurocontrol ja que només és necessari un fitxer per cada procés de construcció del graf.

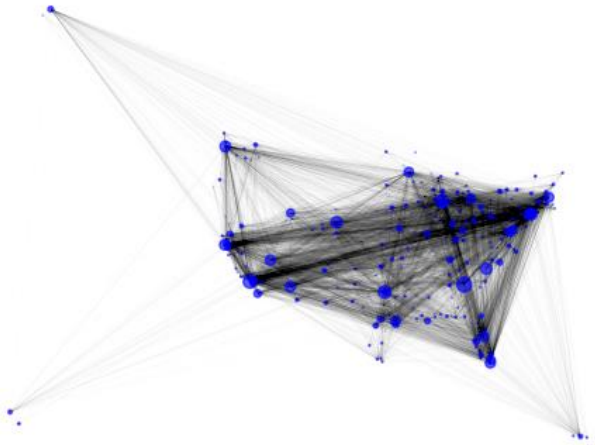

Figura 3.4 Graf de la xarxa d'Estats Units

3.1.4 Graf global

Finalment, la combinació del graf d'Eurocontrol amb el graf d'Estats Units, i juntament amb tots els vols intercontinentals entre ambdós, formen el graf global.

Figura 3.4 Graf de la xarxa global.

3.2 Comparació de xarxes aeroportuàries

Un cop generats els tres grafs, podem estudiar amb detall les seves característiques, així com comparar les propietats de cadascun d'ells. A continuació s'analitzen i les semblances i diferències dels tres grafs obtinguts.

Taula 3.1 Comparativa de les propietats dels tres grafs estudiats.

	Eurocontrol	Estats Units	Global
Mida	23.171	22.343	48.861
Ordre	509	400	909
Diàmetre	5	4	6
Distància mitja	1,960	1,796	2,142
Clustering	0,633	0,700	0,640
Grau mitjà	91,046	111,715	107,506
Grau mínim	1	1	1
Grau màxim	333	300	454
Correlació de grau	-0,168	-0,148	-0,122
Densitat de connexió	0,179	0,280	0,118
Índex Q	-73,756	-77,404	-84,607
Índex Wiener	253.389	143.246	6.363.667

Índex Wiener amb pes	301.577	163.439	334.474.633
Índex d'Estrada	$3{,}243 \cdot 10^{67}$	$9{,}943 \cdot 10^{70}$	$1{,}953 \times 10^{77}$
Índex d'Estrada amb pes	585,504	490,380	928,362
Histograma de graus Eurocontrol	[17, 16, 10, 9, 8, 7, 7, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]		
Histograma de graus Estats Units	[11, 10, 6, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]		
Histograma de graus Global	[29, 28, 13, 12, 11, 10, 10, 9, 9, 8, 8, 8, 8, 8, 8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,		

	0, 0]

Les propietats més bàsiques analitzades són la mida, l'ordre i el diàmetre. La mida del graf d'Eurocontrol és una mica més gran que la mida del graf d'Esats Units, mentre que l'ordre és diferencien en més de 100 nodes. El diàmetre, que es defineix com el camí més llarg d'un determinat node de la xarxa fins l'altre node extrem de la xarxa, és una unitat menor al graf d'Estats Units que d'Eurocontrol. Per tant, la xarxa aeroportuària d'Estats Units està mes ben comunicada a nivell del nombre de salts per arribar a qualsevol altre node. El graf global presenta una mida i ordre fruit de la suma dels anteriors. La mida, però, és major a la suma de les dues mides anteriors ja que s'han afegit nodes els quals l'origen d'un vol resideix a un continent i la destinació a l'altre.

El coeficient de *clustering* més gran pertany al graf d'Estats Units, un valor considerablement alt tenint en compte que l'elevat cost de mantenir una ruta aèria operativa. Per contra, com més gran sigui aquest coeficient, més facilitats tindrà un usuari d'aquesta xarxa per desplaçar-se a un aeroport veí.

Contràriament a les expectatives, el nombre mitjà de connexions per node de la xarxa global és menor al d'Estats Units i superior al d'Eurocontrol. Però per altra banda, la xarxa global és la que té el node amb més connexions de les tres xarxes, denotant un valor de 454 enllaços.

La correlació de graus proporciona informació sobre com es relacionen els nodes entre ells. Les tres xarxes tenen una correlació negativa; denotant que els enllaços dels grafs són heterogenis; connectant nodes de gran importància amb nodes que en tenen menys. La xarxa d'Eurocontrol és la que conté més enllaços entre nodes de diferent rellevància, mentre que la xarxa global és la que es distingeixen més connexions entre nodes de similar importància.

En la següent taula s'especifiquen els nombres de Wiener obtinguts i els seus valors límit.

Taula 3.2 Comparativa dels índex de Wiener pels grafs estudiats.

Xarxa	Límit inferior	Índex Wiener	Límit superior	% Respecte límit superior
Eurocontrol	$1,292 \cdot 10^5$	$2,533 \cdot 10^5$	$2,197 \cdot 10^7$	1.153 %
Estats Units	$7,980 \cdot 10^4$	$1,432 \cdot 10^5$	$1,066 \cdot 10^7$	1.343 %
Global	$2,595 \cdot 10^6$	$6,363 \cdot 10^6$	$1.972 \cdot 10^9$	0.323 %

Com es pot observar, els índex de Wiener calculats tendeixen a estar molt més propers al límit inferior que al límit superior. Al igual que el paràmetre densitat de

connexió, aquest fet denota que els tres grafs formen xarxes poc saturades, en relació al seu nombre de nodes i enllaços.

L'altre índex de notòria rellevància és l'índex Q. Els tres resultats obtinguts d'aquesta propietat són negatius, fet que denominarem com a xarxes rígides. Com detallat anteriorment en el punt 2.2.10, els nodes estan situats en el pla bidimensional en un espai concret i el enllaços poden creuar-se ja que usen l'espai tridimensional i no col·lisionen. Aquest fet determina que les connexions entre els aeroports són eficients. La xarxa global és la que presenta un índex Q més eficient en termes de comunicació entre nodes.

L'últim punt important a tenir en compte, és la distribució de graus de les tres xarxes. La distribució de graus permet conèixer el nombre de nodes que té un determinat grau, és a dir, quants nodes tenen un nombre d'enllaços concret. La figura 3.7 exemplifica la distribució de llei per les tres xarxes. Per graus petits es troben un conjunt elevat de nodes, mentre que al augmentar el nombre de connexions, disminueix paulatinament el nombre de nodes.

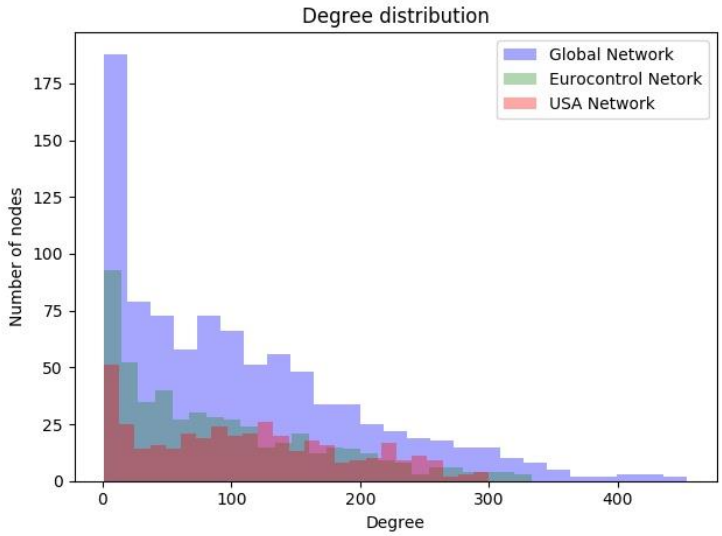

Figura 3.6 Distribució potencial de llei dels graus de les xarxes estudiades.

3.3 Comparació xarxa aeroportuària d'Estats Units (2001-2018)

Aquest punt té per objectiu comparar, breument, l'evolució de la xarxa aeroportuària d'Estats Units entre els anys 2001 i 2018.

Taula 3.3 Comparativa de les propietats del graf d'Estats Units (2001-2018)

	Estats Units (2001)	**Estats Units (2018)**
Mida	2.072	22.343
Ordre	295	400
Diàmetre	5	4
Distància mitja	2,459	1,796
Clustering	0,705	0,700
Grau mitjà	14,074	111,715
Grau mínim	1	1
Grau màxim	132	300
Correlació de grau	-0,416	-0,148
Índex Q	-17,414	-77,404
Índex Wiener	106.656	143.246
Índex Wiener amb pes	$97{,}614 \cdot 10^{6}$	163.439
Índex d'Estrada	$3{,}141 \cdot 10^{18}$	$9{,}943 \cdot 10^{70}$
Índex d'Estrada amb pes	298,722	490,380

La xarxa aeroportuària d'Estats Units ha experimentat, al llarg d'aquets últims 18 anys, un creixement significatiu en nombre d'aeroports, incrementant més d'un centenar els aeroports operatius. La mida de la xarxa, s'ha vist multiplicada per un factor 10 aproximadament. Aquests dos fets repercuteixen directament en altres propietats com la distància mitja i diàmetre, que disminueixen, en canvi, el nombre mitjà de nodes i grau màxim, augmenten.

Un indicador que prova el creixement de la xarxa, és a la correlació de grau, que ha incrementat 0,3 dècimes, fet que indica que les connexions tendeixen estar configurades entre aeroports més grans, ja que tots els nodes han incremenat el nombre de destinacions i vols anuals.

L'índex Q ha augmentat més del quadrúple, i en indicant que l'eficiència de la distribució dels enllaços de la xarxa és bona perquè, donat que els nodes són fixes, el enllaços interconnecten els nodes usant l'espai tridimensional.

L'única propietat que no s'ha vist alterada practicament des del 2001 és el coeficient d'agrupament. Tot i augmentar els nodes i enllaços, l'increment s'ha produit proporcionalment en tota la xarxa.

Podem concloure que la xarxa aeroportuària d'Estats Units és una xarxa que presenta unes característiques sòlides, robustes i de creixement lent, que durant aquest període ha evolucionat positivament cap una xarxa més eficient sempre mantenint o canviant harmònicament les propietats inicials.

4 Estudi de la fallada en cascada

4.1 Fallada en cascada d'una xarxa (cascading failure)

La fallada en cascada d'un sistema interconnectat o d'una xarxa consisteix en un successiu error dels nodes, produït per la fallada d'un determinat node o conjunt de nodes. Com que el sistema ha de compensar els components que fallen, és necessari repartir la càrrega. Un dels principals models d'estudi considera que la càrrega dels nodes que fallen es reparteix equitativament entre els nodes adjacents. Aquests, però, poden sobrecarregar-se i eventualment, fallar també. El procés continua fins que la càrrega pot ésser suportada pels nodes restants, o fins que tots els nodes de la xarxa fallen.

En estudis previs, s'ha demostrat que xarxes del món real, com la xarxa de transport i la xarxa d'Internet, són robustes davant d'errors aleatoris però susceptibles davant d'atacs intencionats. Encara que si l'error comença de forma local, depenent de l'estructura de la xarxa, hi ha nodes altament vulnerables que, si fallen, provocarien que la xarxa fallés globalment. Per aquesta raó, és important identificar eficientment i en el menor temps possible els punts mes dèbils d'una xarxa i prendre accions per reforçar-los. Una possible via seria crear un sistema de seguretat que permetés que la xarxa treballés fins un determinat nivell o càrrega màxima predefinida, deixant un marge suficient per fer front a una possible càrrega extra dels nodes veïns que fallen. Treballant sota aquestes circumstàncies s'asseguraria el correcte funcionament de la xarxa, tot i que ineficient.

Al llarg de la història, hi ha hagut molts casos en que una xarxa s'ha col·lapsat. Un exemple n'és l'apagada de la costa est d'Estats Units l'any 1965. En aquest cas, el país va experimentar per primera vegada una apagada elèctrica que va paralitzar durant 14 hores el ritme habitual d'activitats en vuit estats de la costa est, incloent-hi la ciutat de Nova York. La causa d'aquest col·lapse va ser un error humà, el qual es va produir dies abans de l'apagada, quan el personal de manteniment a configurar de forma incorrecta un relé de protecció en una de les línies de transmissió. En l'àmbit aeronàutic, una de les fallades en cascada més rellevants va succeir l'any 2010. L'espai aeri Europeu va veure's altament compromès des del dia 14 d'abril fins el dia 22 d'abril per l'erupció del volcà Islandès Eyjafjallajökull, que va expulsar cendres volcàniques a l'atmosfera, extenent-se per una àrea de milers de kilòmetres quadrats. Aquest fet va repercutir en més de 100.000 vols durant la setmana afectada, que representa un 48% del tràfic previst en els vuit dies de crisi, i arribant a un màxim del 80% el 18 d'Abril. Conseqüentment, 10 milions de passatgers no van arribar a les seves respectives destinacions.
A més a més, es van afegir 5000 vols per part de les companyies aèries regulars i charter amb els següents motius: per reposicionar els avions, per distribuir les tripulacions; i per accelerar la repatriació dels passatgers afectats. Aquest increment de vols no s'ha tingut en compte en aquest estudi.
Pel que fa al recompte de països més afectats, Finlàndia, Irlanda i Gran Bretanya, a part de Islàndia, van experimentar una reducció del 90% del tràfic

aeri durant 5 dies consecutius. El tràfic intercontinental, afectat també ja que les rutes sobrevolen les immediacions d'Islàndia a diari, es va reconduir en gran part augmentant la latitud de vol al Nord del volcà. En canvi, el tràfic operat per companyies *low-cost* va ser el més afectat de tots, ocasionant un 61% de cancel·lacions durant els dies de crisi. L'elevada exposició i un model de negoci menys flexible contribueix a aquest fet. L'aviació de negoci, menys afectada, amb un 34% de cancel·lacions.

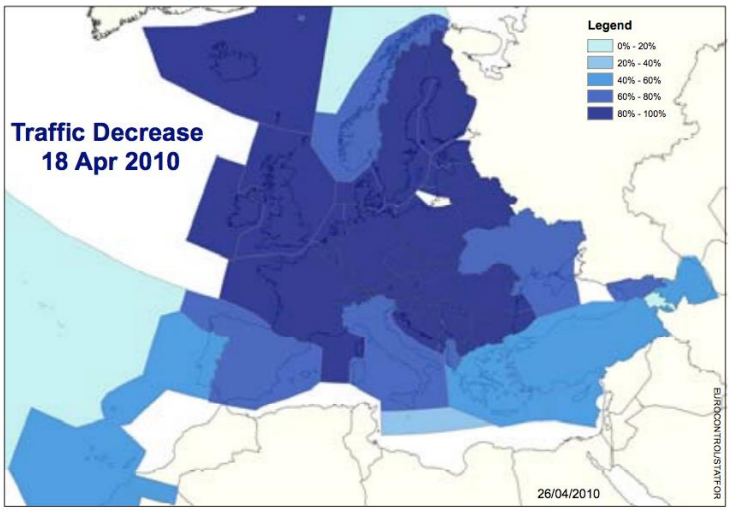

Figura 4.1 Mapa d'afectacions del volcà Eyjafjallajökull el dia 18/04/2010.

La figura 4.1 mostra les afectacions de les mesures adoptades per Eurocontrol per fer front al núvol de cendres. La zona de màximes reduccions del tràfic aeri (zona blau marí) es concentren entre l'Europa central i l'Europa nord, abarcant un radi aproximat de 750km amb centre a l'aeroport de Frankfurt

Prenent com a referència principal l'erupció del volcà Eyjafjallajökull i la importància i afectació que va tenir en la xarxa de transport aeri mundial, en el present document s'estudia la fallada en cascada de la xarxa global en un hipotètic cas de tancament d'un conjunt d'aeroports.

4.2 Escenaris estudiats

L'estudi es centra en analitzar com es distribueix la càrrega enfront un possible tancament d'un conjunt d'aeroports, sigui per una causa natural com va ser el volcà Eyjafjallajökull, o bé per altres motius de seguretat, logística, etc. El subjecte d'estudi per realitzar l'anàlisi de la fallada en és el graf global, compost per el conjunt de grafs d'Eurocontrol i d'Estats Units. Es realitzen diverses simulacions amb l'objectiu de detectar quins nodes són els més susceptibles de la xarxa i en quines condicions la xarxa no és capaç d'absorbir aquesta càrrega, provocant l'error total.

Es defineixen dos paràmetres principals: el radi d'error, R_i, i el percentatge de tolerància de vols que admeten els aeroports, λ. El primer paràmetre determina els nodes inicials de la xarxa que fallaran si es compleix la condició que la distància entre el node estudiat i un aeroport sigui menor al radi d'error. Els valors d'estudi són quatre: 200km, 400km, 600km i 750km. Per altra banda, el percentatge de tolerància estableix la capacitat d'absorció d'un node, anomenada C_{abs}. Aquest percentatge varia des del 5% fins al 35% i s'aplica al nombre de vols diaris de cada aeroport.

$$C_{abs} = \frac{\lambda}{100} \cdot DF_i \quad \textbf{(4.1)}$$

On DF_i és el nombre de vols diaris d'un node i.

Quan un node falla, l'aeroport intenta transferir els seus vols diaris entre els seus nodes adjacents, que accepten més o menys vols en funció de λ. A mesura que un node accepta vols, el llindar C_{abs} disminueix fins, eventualment, arribar a zero en cas que es transfereixin tots els vols possibles. Com més vols diaris tingui un node, més capacitat d'absorció tindrà si falla algun dels seus nodes veïns. En l'apartat 4.3 es detalla l'algorisme usat i el procediment d'actuació al realitzar les transferències de vols.

4.3 Mètode de simulació

Previ a l'algorisme d'estudi de fallada en cascada, és necessari tenir emmagatzemada la informació obtinguda a través dels fitxers obtinguts segons el punt 3.1.1 en el graf estudiat. Cada node del graf global conté el codi icao, el codi iata, el tipus d'aeroport, el país, el continent, el nombre de vols annuals i les seves coordenades. Addicionalment, s'afegeixen els paràmetres: *transfered, cascadeFailure i checked*, que són variables durant el procés de simulació. La informació del primer node de la xarxa, l'Aeroport d'Amsterdam Shiphol és la següent:

- 'icao': 'EHAM'
- 'iata': 'AMS'
- 'type': 'large_airport'
- 'country': 'NL'
- 'continent': 'EU'
- 'annualFlights': 508494
- 'lat': 52.3086013794
- 'lon': 4.763889789579999
- 'transfered': 0
- 'cascadeFailure': 'False'
- 'checked': 'False'

Referent a l'origen i destinació dels vols anuals (*annualFlights)* entre els veïns d'un node, el graf conté el conjunt d'enllaços W amb pes. Cada enllaç *(e)* d'aquest conjunt relaciona els dos aeroports amb el pes que té la ruta, que és el nombre de vols anuals entre ells dos.

Dels tres paràmetres variables de cada node, *cascadeFailure* indica si aquest node falla o no. Un node pot fallar si es configura que inicialment falli, o bé perquè no pot admetre la suficient càrrega de vols que li arriba d'un altre node. El paràmetre checked indica, en cas que un node falli, si ja s'ha simulat la fallada en cascada per aquest aeroport. L'últim atribut variable és *transfered*, que indica el nombre de vols que un altre node li ha distribuït.
A continuació es detalla el procés que es segueix quan un o diversos nodes de a xarxa fallen.

Taula 4.1 Nomenclatura bàsica del procés que simula una fallada en cascada.

N	Nombre total de nodes
Γ_{fail}	Conjunt de nodes que fallen inicialment
k_i	Grau o nombre de veïns del node i
Γ_i	Conjunt format pels nodes que són veïns del node i
AF_i	Nombre de vols anuals que té el node i
$\mathrm{DF}_i = \mathrm{AF}_i/365$	Nombre de vols diaris que té el node i
F_{ij}	Nombre de vols diaris entre dos nodes i, j
T_i	Nombre de vols transferits al node i quan el seu veí j $(j \in \Gamma_i)$ falla
R_i	Radi d'afectació al voltant d'un node i
λ	Percentatge de tolerància de vols diaris que accepta el node i. Es fixa aquest paràmetre com una variable, que pot tenir valors enters des del 5% fins al 35%.
$Cmax_i = \lambda \mathrm{DF}_i - T_i$	Màxim increment efectiu de vols que admet un node i. A mesura que avança la simulació i altres nodes li transfereixen vols a i, aquest paràmetre va disminuint
$Copt_i = 0.5 \cdot Cmax_i$	Increment òptim de vols que admet un node i
$\Gamma\mathrm{asc}_i$	Conjunt de nodes veïns a i ordenats de forma ascendent pels vols que admeten
$\Gamma\mathrm{desc}_i$	Conjunt de nodes veïns a i ordenats de forma descendent pels vols que admeten
$CF_{\Gamma_{fail}}$	Mida de l'allau en fallar el conjunt de nodes atacats
POF	Probability of failure. És el paràmetre que mesura la robustesa de tota la xarxa

L'estudi de la xarxa parteix del graf global i s'analitza la fallada en cascada de cada node en funció de dos paràmetres: el radi d'afectació i el percentatge d'admissió. Els passos que segueix la simulació de fallada en cascada són els següents:

1. Es busquen els aeroports propers al node i que disten menys que R_i. La fórmula de Haversine calcula la distància entre dos nodes en donades les seves coordenades (λ_1, φ_1) i (λ_2, φ_2):

$$a = 2R_T \arcsin\left(\sqrt{(\sin\frac{\varphi_2-\varphi_1}{2})^2 + \cos\varphi_1 \cos\varphi_2 \,(\sin\frac{\lambda_2-\lambda_1}{2})^2}\right) \qquad \textbf{(4.2)}$$

   On $R_T = 6371km$, el radi de la Terra.

2. Es forma el conjunt de nodes Γ_{fail} a partir de la distància entre els nodes. Aquest conjunt engloba els nodes que fallen inicialment, inclòs el node i. Es canvia en tots els nodes d'aquest conjunt el booleà *cascadeFailure* a 'True', per identificar que fallen.

3. Per un determinat node que falli, i, i no s'hagi estudiat prèviament *('checked' = 'False'*), identifica els seus veïns (Γ_i) que estan operatius i per cada un d'ells, identifica el nombre de vols diaris entre i, j (F_{ij}). Es transfereix F_{ij} al node j en funció de la seva capacitat òptima ($Copt_j$) si:

$$\mathrm{F}_{ij} < Copt_j \qquad \textbf{(4.3)}$$

   En cas que j no pugui admetre la totalitat de F_{ij} però sí una part de F_{ij} ($\mathrm{F}_{ij} > Copt_j$ essent $Copt_j \,! = 0$), es transfereixen únicament aquests vols que pot admetre, arribant j al seu màxim d'admissió sense provocar que falli.
   Cal remarcar que aquesta capacitat sempre és la meitat de la capacitat total d'un node, $Cmax_j$. Per tant, $Copt_j$ és la capacitat òptima que satisfà que es distribueixin el més equitatiu possible els vols d'un node a l'hora de ser transferits entre els seus veïns.

4. Si no s'aconsegueixen transferir tots els vols de i entre els seus veïns operatius, es transferiran els vols que resten al primer veí operatiu de $\Gamma\mathrm{desc}_i$ (el node que tingui més capacitat d'admissió) o $\Gamma\mathrm{asc}_i$ (el node que tingui menys capacitat d'admissió), provocant que aquest falli.

5. En cas de que sí s'hagin pogut transferir els vols entre els veïns sense forçar l'error, s'analitza el següent node que falla.

6. Un cop estudiats tots els nodes de la xarxa per tots valors del paràmetre λ, s'escriu un fitxer d'extensió '.txt' que en cada línia conté el node que

falla inicialment, λ, $CF_{\Gamma_{fail}}$ i Γ_{fail} (el nombre de nodes que fallen inicialment).

7. Un cop generat aquest fitxer de resultats, es calcula el percentatge de fallada, POF, per cada λ:

$$POF = \frac{CF_{\Gamma_{fail}}}{N-\Gamma_{fail}} \cdot 100 \quad \textbf{(4.3)}$$

On POF determina la robustesa de la xarxa. Quan $POF = 100$ indica que en fallar el conjunt Γ_{fail}, tota la resta de nodes han fallat. En canvi, $POF = 0$, significa tot i fallant els nodes de Γ_{fail}, no ha afectat al correcte funcionament de la xarxa. Per tant, quan més petit és POF menys vulnerable és la xarxa.

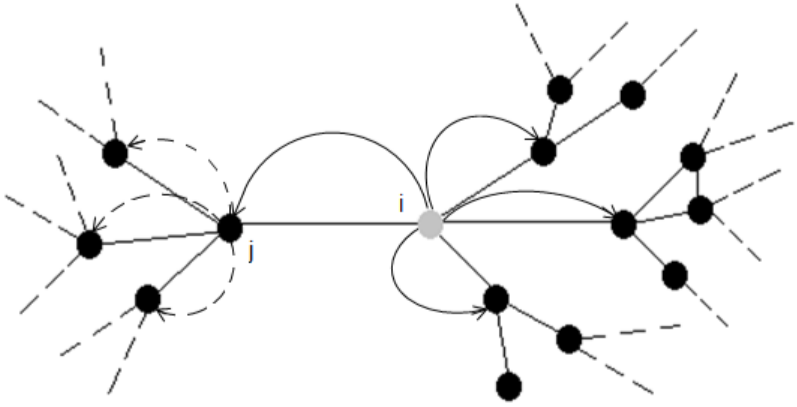

Figura 4.2 Distribució de la càrrega del node que falla entre els veins.

Per realitzar les simulacions s'ha usat el paquet *NetworkX* de *Python*, que permet crear i gestionar i estudiar xarxes complexes.

4.4 Estratègies d'atac

Per un determinat radi i percentatge de tolerància i amb la finalitat d'observar el comportament que segueix la xarxa global quan un node no pot transferir tots els seus vols, es simulen dues estratègies a l'hora de forçar l'error en un dels seus veïns:

- Atac al node més petit (LL): consisteix en seleccionar l'aeroport operatiu més petit al qual està connectat al node que falla. Aquest serà l'encarregat

de rebre tots els vols que l'altre node no ha pogut transferir. Conseqüentment, aquest node falla ja que no té la suficient capacitat d'absorció. El conjunt Γ_{asc} representa els nodes veïns, que d'entre aquests, s'escollirà el que tingui menys C_{abs}.

- Atac al node més gran (HL): és el procés contrari a l'altra estratègia d'atac. Es basa en provocar la l'error al node veí més gran amb el que connecta. El conjunt Γ_{desc} representa els nodes veïns, que d'entre aquests, s'escollirà el que tingui més C_{abs}.

5 Anàlisi dels resultats

En aquest apartat, es procedirà a analitzar els resultats obtinguts de les simulacions realitzades per estudiar la fallada en cascada en la xarxa global. S'han realitat simulacions pels dos paràmetres bàsics: el percentatge de tolerància, comprès entre 5% i 35%, i el radi d'afectació inicial des de 200km fins a 750km. Aquest últim pretén simular el més versemblant possible els efectes d'una hipotètica erupció d'un volcà que causi afectacions de la mateixa escala que el volcà Eyjafjallajökull. Per cada radi estudiat, s'han efectuat els dos tipus d'atacs plantejats anteriorment, a l'hora de forçar l'error en un node.

Tot i que es considera que una xarxa falla completament quan, degut a un error inicial no es distribueix la càrrega d'aquests nodes provocant que fallin la totalitat dels altres nodes de la xarxa, s'ha establert un llindar del 85% a partir el qual es determina que una xarxa falla globalment. Tenint en compte que la xarxa global esta constituïda per 867 nodes, el llindar es fixa a 737 nodes. També és important tenir present que la xarxa global comprèn els aeroports tant d'Eurocontrol com d'Estats Units, fet que no és comprensible transferir vols intracontinentals a un aeroport ubicat a l'altre continent, convertint-lo en un vol intercontinental. En canvi, per els vols de caire intercontinental sí és beneficiós estudiar les possibles repercussions que pugui tenir una fallada en cascada de grans dimensions.

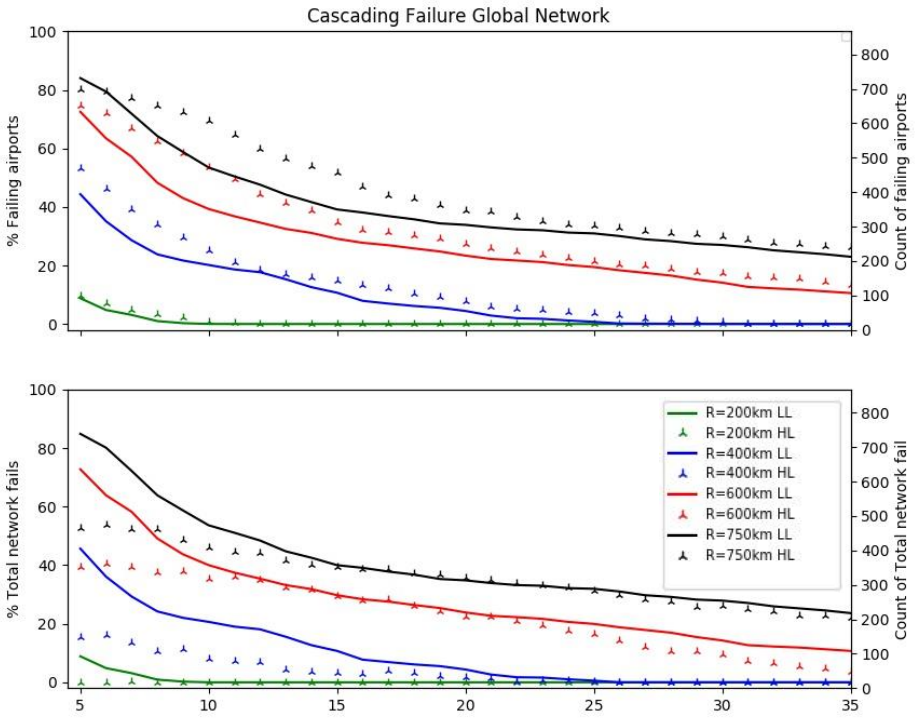

Figura 5.1 a) Percentatge de nombre d'aeroports que fallen. b) Percentatge de fallades totals d'entre totes les simulacions.

La figura superior representa el percentatge d'aeroports que fallen a tota la xarxa, independentment de si provoquen una fallada total o no. La figura inferior mostra el percentatge de simulacions que ha fet fallar la xarxa globalment. Els dos tipus d'atacs plantejats han estat: atac al node més petit i atac al node més gran. La tendència que segueixen totes les línies de les dues figures és disminuir els nodes que fallen al llarg del percentatge d'admissió, més o menys ràpid en funció del radi d'afectació.

Analitzant la primera figura, per a petites toleràncies (fins a 12%), les simulacions en què s'ha usat el l'atac de defallir el node més important primer (HL), repercuteixen en un major nombre de nodes de la xarxa; provocant la fallida de més aeroports que si s'usa l'atac LL. Resulta fàcil, y lògic, relacionar el fet que fallin més aeroports d'una xarxa amb que hi haurà un percentatge major de fallades totals a aquesta. La figura 5.1b mostra que, per atacs HL, el nombre de simulacions en les quals han conduit a la fallada total, és quasi bé la meitat que per atacs LL. Aquest fet s'observa pels radis $R_i = 750km, R_i = 600km$ i $R_i = 400km$, que, aquest últim, fins i tot la diferència respecte l'atac LL és major que la meitat.

Pel $R_i = 200km$ s'obté poca informació, ja que les distàncies entre els aeroports són, en la majoria dels casos majors a aquest radi. Per tant, els nodes induïts per $R = 200km$ seran minoritaris respecte els altres radis. L'afectació més gran que pot arribar a generar s'observa per $\lambda = 5\%,$ i representa menys del 10% de simulacions amb fallada global. A més a més, a partir d'aquest percentatge de tolerància, deixa de tenir efecte aquesta simulació perquè ja no provoca cap fallada de nodes.

Per toleràncies mitjanes fins a 25%, el nombre de simulacions amb fallada global (5.1b) es pràcticament el mateix atacant amb l'estratègia HC com LL. Quan el percentatge és major al 25%, torna a succeir el mateix resultat que per toleràncies petites però a menor escala, ja que els nodes són capaços de reduir considerablement el nombre d'aeroports que fallen al poder acceptar més transferències de vols.

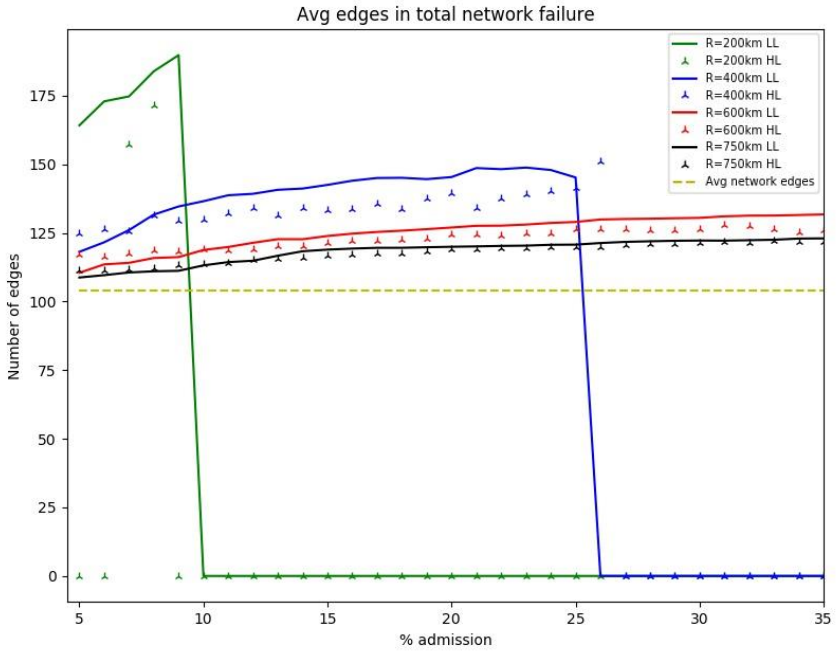

Figura 5.2 Mitjana d'enllaços dels nodes inicials que indueixen una fallada total.

Per tal d'identificar quins són els nodes més vulnerables de la xarxa, s'ha elaborat aquest gràfic. Com es pot observar, estan representats el nombre mitjà d'enllaços del conjunt de nodes inicials que provoquen una fallada total de la xarxa (Γ_{fail}). Quan el nombre d'enllaços és zero, indica que la xarxa ha pogut distribuir tots els vols i no falla totalment.

Aquesta figura exemplifica la tendència que segueixen el nombre d'enllaços dels nodes que formen el conjunt Γ_{fail}. Com la figura mostra, tendeix a estabilitzar-se al augmentar el radi. Inicialment, per un radi $R = 200km$, i per atacs HL i LL, descriuen una trajectòria que ràpidament tendeix a zero. Els altres radis mostren que, per a λ petites, el nombre d'enllaços mitjà és bastant similar, aproximadament entre 105 i 120 enllaços, depenent del radi estudiat. A mesura que augmenta la tolerància, totes les simulacions denoten l'increment, molt lleuger, del nombre mitjà d'enllaços. En tots els casos, el conjunt de nodes que defallen inicialment i provoquen una fallada global de la xarxa tenen entre 105 i 140 enllaços cadascun. A més a més s'observa que com més gran és el radi, aquest nombre s'acosta més a la mitjana d'enllaços mitjà de la xarxa. Per últim, tots els atacs LL presenten un nombre d'enllaços lleugerament inferior al seu homòleg HL.

Els resultats obtinguts són suficientment determinants per demostrar que els possibles efectes d'un volcà similar al islandès podrien posar en perill la xarxa de transport aeri tant d'Estats Units com d'Europa.

6 Conclusions

En la nostra vida quotidiana hi ha moltes xarxes complexes de les que hi formem part o hi participem rutinàriament, com per exemple el sistema de telecomunicacions, la xarxa elèctrica, l'Internet o xarxes de transport de béns. És molt important comprendre amb el màxim detall possible com evolucionen i quins són els paràmetres que poden modificar de forma ràpida el seu comportament, per tal de desenvolupar xarxes d'alta tolerància, que són les menys afectades envers atacs intencionats o errors aleatoris.

Per poder dur a terme aquest estudi, s'han generat a partir de dades públiques tres xarxes: la xarxa formada pels aeroports dels estats membres d'Eurocontrol, la xarxa d'aeroports d'Estats Units i, una xarxa global, que compren les dues anteriors. S'ha estructurat el treball en dues parts. La primera ha consistit en generar i estudiar les propietats generals de les tres xarxes, com l'ordre, la mida, el coeficient d'agrupament, etc. Hem pogut observar que, tot i que la xarxa d'Estats Units és lleugerament més petita en nombre de nodes i de connexions que la d'Eurocontrol, el nombre mitjà de graus i el coeficient d'agrupament són superiors. També ho és el coeficient de Wiener, fet que determina que la xarxa d'Estats Units és la més densa de les tres. L'índex Q corrobora que la xarxa d'Estats Units és més eficient que la xarxa d'Eurocontrol.

En la segona part del present estudi, prenent com a referència l'erupció del volcà Eyjafjallajökull l'any 2010, s'ha establert l'algorisme que simula la propagació de fallades en cascada per la xarxa global, al forçar un error inicial a uns determinats aeroports. Les principals variables de l'algorisme són el radi d'afectació, la tolerància dels aeroports a l'hora de rebre la càrrega d'altres nodes veïns i el tipus d'atac per induir un node a l'error. Per un determinat radi i una tolerància menor al 12%, les simulacions amb estratègia d'atac HL afecten a un nombre major d'aeroports que l'atac LL, però les conseqüències de fallada en cascada a nivell global són significativament menors. Per simulacions amb tolerància gran, major al 22%, es produeix el mateix resultat que per toleràncies petites però en menor mesura, ja que els aeroports són capaços d'absorbir més nombre de vols sense fallar. A partir de l'anàlisi dels aeroports que fallen inicialment i provoquen una fallada total, s'han obtingut resultats suficient consistents per afirmar que els aeroports més vulnerables de la xarxa global són els de tamany mitjà, amb un nombre de connexions amb altres aeroports suaument superior respecte la mitjana de connexions de la resta d'aeroports.

Donada la importància actual del transport aeri i la vulnerabilitat en determinades situacions, aquest estudi pot ajudar en el procés de millora de la infraestructura aèria a l'hora d'evitar fallades que, eventualment, podrien ser catastròfiques.

7 Referències

[1] Halvin, S., Cohen, R.: Complex Networks. Structure, Robustness and Function. (Cambrige University Press. Nova York. 2010) ix, 9

[2] Estrada,E.: The structure of complex networks. Theory and applications. (Oxford University Press Inc. Nova York. 2011)

[3] Wilson, R.J.: Introduction to Graph Theory. (Longman Group Ltd. 1979)

[4] Bondy, J., Murty, U.: Graph theory with applications. (Elseiver Science Publishing Co.,Inc. Nova York. 1976)

[5] Newman, M. (2010). Networks An Introduction. University of Michigan and Santa Fe Institute: Oxford University.

[5] Ruohonen, K. (2018). Graph Theory. Retrieved from http://math.tut.fi/~ruohonen/GT_English.pdf

[6] Harrigan, M. (2018). A Network Analyst's View of the Block Chain. Retrieved from https://coindesk.com/network-analysts-view-block-chain/

[7] Open Flights. Airport Information https://openflights.org/data.html

[8] Statistics and Forecasts (STATFOR) Eurocontrol. https://www.eurocontrol.int/statfor

[9] ATADS Airport Information Federal Aviation Administration. https://aspm.faa.gov/opsnet/sys/main.asp

[10] Schafer, M., Strohmeier, M., Smith, M., Fuchs, M., Pinheiro, R., Lenders, V., & Martinovic, I. (2018). OpenSky Report 2016: Facts and Figures on SSR Mode S and ADS-B Usage [Ebook]. University of Kaiserslautern, Germany. Retrieved from https://www.cs.ox.ac.uk/files/8371/dasc2016.pdf

[11] E. Estrada, (2012). Complex networks in the Euclidean space of communicability distances. University of Strathclyde

[12] M. S. Campbell, B. (2018). Volcanoes in European history: Exploring Environmental History Podcast. Retrieved from http://activehistory.ca/2010/04/volcanoes-in-european-history-exploring-environmental-history-podcast/

[13] Zhang, W., Pei, W., & Guo, T. (2018). An efficient method of robustness analysis for power grid under cascading failure. Retrieved from https://core.ac.uk/download/pdf/82262132.pdf

[14] Newman, M. E. J. : Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. The American Physical Society. Rev. E **64** (016132). (2001)

[15] Comellas, F. <<Models deterministes de xarxes complexes>>. Butll.De La SCM, 22(1) (2007), 23-43

[16] Bonacich, P. : Power and Centrality: A Family of Measures. University of Chicago. **95**(5) 1170-82. (1987)

[17] Modelling the Air Transport with Complex Networks: a short review. (2018). [Ebook]. Polytechnic University of Madrid. Retrieved from https://arxiv.org/pdf/1302.7017.pdf

[18] Barabasi, A. (2018). Scale Free Networks. Retrieved from http://barabasi.com/f/124.pdf

[19] IATA Airport Code – Wikipedia free encyclopedia (01-05-2018) https://en.wikipedia.org/wiki/IATA_AirportCode

[20] T Bullmore, E., & Sporns, O. (2018). Complex brain networks: Graph theoretical analysis of structural and functional systems [Ebook]. Nature Reviews Neuroscience. Retrieved from https://www.researchgate.net/profile/Olaf_Sporns/publication/23974889_Complex_brain_networks_Graph_theoretical_analysis_of_structural_and_functional_systems/links/56a2476b08ae1b65112c86e9/Complex-brain- networks-Graph-theoretical-analysis-of-structural-and-functional- systems.pdf

ANNEX

TÍTOL DEL TFG: Analysis of the European airport network as a complex network

TITULACIÓ: Grau en Enginyeria d'Aeronavegació

AUTOR: Pol Gelabert Goday

DIRECTOR: Francesc Comellas Padró

DATA: 7 de setembre del 2018

```
# -*- coding: utf-8 -*-
import os
import sys
import sip
#import PyQt5
#import PySide2
import PyQt4



try:
    import matplotlib
    matplotlib.use("Qt4Agg")
    #matplotlib.use("Qt5Agg")
    import matplotlib.pyplot as plt
except:
    raise
plt.close('all')
#os.system('cls' if os.name == 'nt' else 'clear')
import cProfile
import pstats



import pylab as P
##from math import exp, log, sqrt
from math import exp, log, sqrt, ceil, floor
import operator
import random
import networkx as nx
import xlrd
from math import exp, log, pi, sin, cos, atan2, sqrt
#from geopy.distance import vincenty
import numpy as np; np.set_printoptions(threshold=np.inf)
import time
from mpl_toolkits.basemap import Basemap
import pandas as pd
import requests
import datetime
import time
import operator



def unixTimestamp2ReadableDate(timen):
    return datetime.datetime.fromtimestamp(int(timen)).strftime('%d-%m-%Y')


# ######################################################################


## per limitar els decimals a 4 en una llista / tuple
## x = map(prettyfloat, x)
```

```
## tambe es pot fer servir round(x,4)
class prettyfloat(float):
    def __repr__(self):
        return "%0.4f" % self

def float2int(value):
    if (value - int(value)) < 0.5: return floor(value)
    else: return ceil(value)

## calcula distancia en km entre dos punts donats en coordenades
# def haversine_distance(loc1,loc2):
#     """Haversine formula"""
#     """- give coordinates as (lat_decimal, lon_decimal) tuples. Returns distance in
        km"""
#     earth_radius = 6371.0 # km
#
#     to_radians = lambda x : x * pi / 180.0
#     lat1 = map(to_radians, loc1[0])
#     lon1 = map(to_radians, loc1[1])
#     lat2 = map(to_radians, loc2[0])
#     lon2 = map(to_radians, loc2[1])



    # haversine formula
    # hdlat = (lat2 - lat1) / 2.0
    # hdlon = (lon2 - lon1) / 2.0
    # a = sin(hdlat)**2 + cos(lat1) * cos(lat2) * sin(hdlon)**2
    # c = 2.0 * atan2(sqrt(a), sqrt(1.0 - a))
    # km = earth_radius * c
    # return km



class Airport:
    """  def __init__(self)= Constructor """
    def __init__(self, icao, iata, type, name, lat, lon, elev, continent, country, region,
         municipality):
        self.icao = icao
        self.iata = iata
        self.type = type
        self.name = name
        self.lat = lat
        self.lon = lon
        self.elev = elev
        self.continent = continent
        self.country = country
        self.region = region
        self.municipality = municipality
        self.annualFlights = 0
        self.annualFlightsRadar = 0


#Manual method to display info:
# print ('{} {}'.format(airp1.id, airp1.name))
```

```
    def airport_info(self):
        return '{} {} {} {} {} {} {} {} {} {} {} {} {}'.format(self.icao, self.iata, self.type, self.name,
         self.lat, self.lon, self.elev, self.continent, self.country, self.region, self.municipality,
         self.annualFlights, self.annualFlightsRadar)

    @classmethod
    def set_id(cls,id_airport):
        cls.id = id_airport

    @classmethod
    def from_string(cls, airp):
        id, ident, name = airp.split('-')
        return cls(id, ident, name)

    def __repr__(self):
        # Per no retornar un tipus Objecte genèric
        return  "Airport info: {} {} {} {} {} {} {} {} {} {} {} {} {}".format(self.icao, self.iata, self.type,
         self.name, self.lat, self.lon, self.elev, self.continent, self.country, self.region,
         self.municipality, self.annualFlights, self.annualFlightsRadar)


    def __str__(self):
        return  "{} {} {} {} {} {} {} {} {} {} {} {} {}".format(self.icao, self.iata, self.type, self.name,
         self.lat, self.lon, self.elev, self.continent, self.country, self.region, self.municipality,
         self.annualFlights, self.annualFlightsRadar)


    #Getter
    @property
    def name_airp(self):
        return "Airport name is: '{}'".format(self.name)

    @property
    def geticao(self):
        return "self.icao is : '{}'".format(self.icao)

    @property
    def loc(self):
        return [self.lat, self.lon]

    #Setter
    @name_airp.setter
    def name_airp(self, name_airport):
        name = name_airport
        self.name = name




# import sys
# old_stdout = sys.stdout
# path_out = 'out/'
# log_file = open(path_out + 'out_Networks.log','w')
```

```
# sys.stdout= log_file



#########################################################
################    AIRPORTS    ####################
#########################################################

path = 'files/worldAirports2.xlsx'
print ('Airports data-> ' + path + ': ' + str(os.path.exists(path)))

wall = xlrd.open_workbook(path)
sheet = wall.sheet_by_index(0)

data = [[sheet.cell_value(r,c) for c in range(sheet.ncols)] for r in range(sheet.nrows)]


dict_airp_all = {}
dict_airp_all_of_them = {}
dict_airp_eu = {}
dict_airp_usa = {}
icao2iata = {}
iata2icao = {}
list_countries = []

dict_airp_NA = {}

for i in range(1,sheet.nrows):
    cord = data[i][11]
    a = cord.split(', ')
    [float(i) for i in a]
    lat_i = a[1]
    lon_i = a[0]

    airport = Airport(data[i][0], data[i][9], data[i][1], data[i][2], lat_i, lon_i, data[i][3],
        data[i][4], data[i][5], data[i][6], data[i][7])

    dict_airp_all_of_them.update({airport.icao: airport})
    icao2iata.update({airport.icao: airport.iata})
    iata2icao.update({airport.iata: airport.icao})

    #if airport.type == "small_airport" or airport.type == "medium_airport" or airport.type
        == "large_airport":
    if airport.type == "small_airport" or airport.type == "medium_airport" or airport.type
        == "large_airport":

        if ((airport.lat== 0 and airport.lon == 0) or airport.icao == "US-0742"): pass          #
         restriccions d'aeroports conflictius (nomes 1).
        else:

            dict_airp_all.update({airport.icao: airport})

            if airport.continent == "EU":
            #if airport.country == "ES":
                dict_airp_eu.update({airport.icao: airport})
```

```
            if airport.country not in list_countries:
                list_countries.append(airport.country)

        #if airport.country == "US":
        #    dict_airp_usa.update({airport.icao: airport})

        if airport.continent =="NA":
            dict_airp_NA.update({airport.icao: airport})


########################################
########## annualFlights EUROPE ######
########################################

month = {1:31, 2:28, 3:31, 4:30, 5:31, 6:30, 7:31, 8:31, 9:30, 10:31, 11:30, 12:31}
path = 'files/Eurocontrol/1_17-18.xlsx'
#start_time = time.time()
print ('Opening Europe AnnualFlights -> ' + path + ': ' + str(os.path.exists(path)))

icao2annualFlights = {}
diff_airp = []
for j in range(1,13):

    path = 'files/Eurocontrol/' + str(j) + '_17-18.xlsx'


    wall = xlrd.open_workbook(path)
    sheet = wall.sheet_by_index(0)

    data = [[sheet.cell_value(r,c) for c in range(sheet.ncols)] for r in range(sheet.nrows)]

    dict_annualFlights = {}

    for i in range(1,sheet.nrows):

        if data[i][0] in dict_annualFlights.keys():
            data1 = float(data[i][3])
            data2 = float(dict_annualFlights[data[i][0]])
            if data1 > data2:
                dict_annualFlights[data[i][0]] = float2int(month[j]*(float(data[i][3])))
        else:
            dict_annualFlights[data[i][0]] = float2int(month[j]*(float(data[i][3])))


    if j == 1: icao2annualFlights = dict_annualFlights
    else:
        for key in list(dict_annualFlights.keys()):
            if key not in icao2annualFlights.keys():                #Elimina els aeroports no
         comuns
                diff_airp.append([key, dict_annualFlights[key]])
                del dict_annualFlights[key]

        for key in list(icao2annualFlights.keys()):                 #Elimina els aeroports no
         comuns
            if key not in dict_annualFlights.keys():
```

```
            diff_airp.append([key, icao2annualFlights[key]])
            del icao2annualFlights[key]

        icao2annualFlights = {k : icao2annualFlights.get(k, 0) + dict_annualFlights.get(k,0)
         for k in (set(icao2annualFlights.keys()) & set(dict_annualFlights.keys()))}
         #Sumatori dels dos dicts per actualitzar les dades


lypr = icao2annualFlights['LYPR']
icao2annualFlights['BKPR'] = lypr
del icao2annualFlights['LYPR']; del icao2annualFlights['ENQV']; del
        icao2annualFlights['ENSE']
del icao2annualFlights['ENLE']
del icao2annualFlights['ENLA']
del icao2annualFlights['ENQG']
del icao2annualFlights['ENSL']
del icao2annualFlights['LERJ']
del icao2annualFlights['ENGA']
del icao2annualFlights['EKDF']
del icao2annualFlights['ENFB']
del icao2annualFlights['ENWS']
del icao2annualFlights['ENHE']
del icao2annualFlights['ENBE']
del icao2annualFlights['ENQS']
del icao2annualFlights['ENUA']
del icao2annualFlights['ENSF']
del icao2annualFlights['EHJA']
del icao2annualFlights['ENVF']
del icao2annualFlights['ENQO']
del icao2annualFlights['ENOA']
del icao2annualFlights['ENQR']
del icao2annualFlights['EHLL']
del icao2annualFlights['ENUB']
del icao2annualFlights['EPLB']
del icao2annualFlights['ENQK']
del icao2annualFlights['EHLF']
dict_airp_all['LTCT'] = dict_airp_all['TR-0026']
dict_airp_all['LTCT'].icao = 'LTCT'
dict_airp_all_of_them['LTCT'] = dict_airp_all_of_them['TR-0026']
dict_airp_all_of_them['LTCT'].icao = 'LTCT'
del dict_airp_all['TR-0026']
del dict_airp_all_of_them['TR-0026']
del icao2annualFlights['LECH']
del icao2annualFlights['ENGC']
del icao2annualFlights['LTFG']
del icao2annualFlights['LTCU']
del icao2annualFlights['LTCB']
del icao2annualFlights['ENOC']
del icao2annualFlights['ENQC']
del icao2annualFlights['ENHM']
del icao2annualFlights['ENUK']
del icao2annualFlights['ENQB']
del icao2annualFlights['ENWV']
del icao2annualFlights['ENUG']
del icao2annualFlights['ENVR']
```

```
del icao2annualFlights['ENXN']
del icao2annualFlights['ENXW']
del icao2annualFlights['EHKP']
del icao2annualFlights['LFSL']
del icao2annualFlights['EKTE']


# Diccionari icao - numero de node del graph eu i usa
j=1
icao2node_eu = {}
for key in icao2annualFlights.keys():
    if dict_airp_all[key].continent == 'EU':
        icao2node_eu.update({key:j})
        j=j+1


########################################
########## annualFlights  USA ########
########################################

d = []
for key in icao2node_eu.keys():
    d.append(key)


# file = open("files/query_EU.txt","w")
# for item in d:
#     file.write("select estdepartureairport, estarrivalairport, day from flights_data4 where
          estdepartureairport='" + item + "' and day>=1483228801 and day<=1514764740
          and estdepartureairport is not null and estarrivalairport is not null;")
#
# file.write("quit;")
# file.close()

########################################
########## annualFlights  USA ########
########################################

path = 'files/USA/USA_ops2.xlsx'
print ('Opening USA AnnualFlights -> ' + path + ': ' + str(os.path.exists(path)))

#diff_airp_usa = []

wall = xlrd.open_workbook(path)
sheet = wall.sheet_by_index(0)
data = [[sheet.cell_value(r,c) for c in range(sheet.ncols)] for r in range(sheet.nrows)]
icao2annualFlights_usa = {}
icao1 = None
col  = 11
for i in range(-1,517):
    if data[9+i][0] == "ADM":
        icao1 == "KADM"
        icao2annualFlights_usa[icao1] = float(data[9+i][col])
    elif data[9+i][0] == "ADS":
        icao1 == "KADS"
```

```
        icao2annualFlights_usa[icao1] = float(data[9+i][col])
    #elif data[i][0] == "APC": icao1 == "KAPC"
    elif data[9+i][0] == "BAK": ##En realitat iata code: GYD icao code: UBBB
        icao1 == "UBBB"
        icao2annualFlights_usa[icao1] = float(data[9+i][col])
    elif data[9+i][0] == "BKV":
        icao1 == "KBKV"
        icao2annualFlights_usa[icao1] = float(data[9+i][col])
    elif data[9+i][0] == "PHL":
        icao1 == "KPHL"
        icao2annualFlights_usa[icao1] = float(data[9+i][col])
    elif data[9+i][0] == "TKI":
        icao1 == "KTKI"
        icao2annualFlights_usa[icao1] = float(data[9+i][col])
    elif i ==528:
        print (data[9+i][0])
        print (data[9+i][col])
#       icao1 == "KBKV"
#       icao2annualFlights_usa[icao1] = float(data[i][14])
    else: icao1 = iata2icao[data[9+i][0]]

    icao2annualFlights_usa[icao1] = float(data[9+i][col])

dict_airp_usa2 = {}
for key in icao2annualFlights_usa:
    dict_airp_all[key].annualFlights = icao2annualFlights_usa[key]
    dict_airp_usa2[key] = dict_airp_all[key]
j=1
icao2node_usa = {}
for key in dict_airp_usa2.keys():
    #if dict_airp_all[key].continent == 'NA' or dict_airp_all[key].continent == 'SA':
    if dict_airp_all[key].continent == 'NA':
        icao2node_usa.update({key:j})
        j=j+1
    else:
        print (key)

#######################################
######### Graph NetworkX EUROPE ####
#######################################
#######################################
#start_time = time.time()


######### Basemap definition #######
fig1 = plt.figure(1)
m = Basemap(projection='mill', lat_0=0, lon_0=0, resolution='c')
m.drawcountries(linewidth = 0.5)
#m.fillcontinents(color='white',lake_color='white')
m.fillcontinents()
m.drawcoastlines(linewidth=0.5)

E = nx.Graph()
E.add_nodes_from(icao2node_eu.values())
```

```
pos_E = {}
lats = []
lons = []

for key in icao2node_eu:
    i= icao2node_eu[key]
    if key in dict_annualFlights:
        E.add_node(i, icao = key, iata = dict_airp_all[key].iata, lon =
         float(dict_airp_all[key].lon), lat = float(dict_airp_all[key].lat), type =
         dict_airp_all[key].type, country = dict_airp_all[key].country, continent =
         dict_airp_all[key].continent, annualFlights = icao2annualFlights[key])
    else: pass
    pos_E[i] = m(E.node[i]['lon'], E.node[i]['lat'])


############### BaseMap definition graph USA #############

m2 = Basemap(projection='mill', lat_0=0, lon_0=0, resolution='c')
m2.drawcountries(linewidth = 0.5)
#m2.fillcontinents(color='green',lake_color='white')
m2.fillcontinents()
m2.drawcoastlines(linewidth=0.5)

m3 = Basemap(projection='mill', lat_0=0, lon_0=0, resolution='c')
m3.drawcountries(linewidth = 0.5)
#m3.fillcontinents(color='green',lake_color='white')
m3.fillcontinents()
m3.drawcoastlines(linewidth=0.5)

#####################################
######### Graph NetworkX USA #######
#####################################
#####################################
#start_time = time.time()


U = nx.Graph()
U.add_nodes_from(icao2node_usa.values())

pos_U = {}
lats = []
lons = []
for key in icao2node_usa:
    i= icao2node_usa[key]
    #if key in dict_annualFlights:
    U.add_node(i, icao = key, iata = dict_airp_all[key].iata, lon =
         float(dict_airp_all[key].lon), lat= float(dict_airp_all[key].lat), type =
         dict_airp_all[key].type, country = dict_airp_all[key].country, continent =
         dict_airp_all[key].continent, annualFlights = dict_airp_all[key].annualFlights)
         #else: pass
    pos_U[i] = m(U.node[i]['lon'], U.node[i]['lat'])


#####################################
######### Graph NetworkX ############
```

```
#####################################
#####################################  U S A + E U R
      ######################

# j=1
# icao2node_E_U_Di = {}
# for key in icao2annualFlights.keys():
#     if dict_airp_all[key].continent == 'EU':
#         icao2node_eu.update({key:j})
#         j=j+1
#
# j=1
# icao2node_usa = {}
# for key in dict_airp_usa2.keys():
#     #if dict_airp_all[key].continent == 'NA' or dict_airp_all[key].continent == 'SA':
#     if dict_airp_all[key].continent == 'NA':
#         icao2node_usa.update({key:j})
#         j=j+1
#
#

# E_U_Di.add_nodes_from(icao2node_usa.values())
#
# pos_E_U_Di_Di = {}
# lats = []
# lons = []
# for key in icao2node_usa:
#     i= icao2node_usa[key]
#     #if key in dict_annualFlights:
#     U.add_node(i, icao = key, iata = dict_airp_all[key].iata, lon =
        float(dict_airp_all[key].lon), lat= float(dict_airp_all[key].lat), type =
        dict_airp_all[key].type, country = dict_airp_all[key].country, continent =
        dict_airp_all[key].continent, annualFlights = dict_airp_all[key].annualFlights)
        #else: pass
#     pos_U[i] = m(U.node[i]['lon'], U.node[i]['lat'])

################### NA
# j=1
# icao2node_NA = {}
# for key in dict_airp_NA.keys():
#     icao2node_NA.update({key:j})
#     j=j+1
#
# NA = nx.Graph()
# NA.add_nodes_from(icao2node_NA.values())
#
# pos_NA = {}
# for key in icao2node_NA:
#     i= icao2node_NA[key]
#     NA[i]['id'] = i
#     NA[i]['icao'] = key
#     NA[i]['iata'] = dict_airp_NA[key].iata
#     NA[i]['coord'] = (float(dict_airp_NA[key].lon), float(dict_airp_NA[key].lat))
#     NA[i]['type'] = dict_airp_NA[key].type
#     NA[i]['country'] = dict_airp_NA[key].country
```

```
#     NA[i]['continent'] = dict_airp_NA[key].continent
#     if key in icao2annualFlights_usa: NA[i]['annualFlights'] =
        icao2annualFlights_usa[key]
#     else: NA[i]['annualFlights'] = 0
#
#     pos_NA[i] = m2(NA[i]['coord'][0],NA[i]['coord'][1])


################################################################################
        ############
################################  RUTES
        ########################################
################################################################################
        ############
# path = 'files/routes.xlsx'
# print (os.path.exists(path))
#
# wall = xlrd.open_workbook(path)
# sheet = wall.sheet_by_index(0)
#
# data = [[sheet.cell_value(r,c) for c in range(sheet.ncols)] for r in range(sheet.nrows)]
#
# count = 0
# icao_orig = ''
# icao_dest = ''
# node_orig = 0
# node_dest =0
# iata =''
# for i in range(1,sheet.nrows):
#     if data[i][2] == "GYD": data[i][2]="BAK"
#     if data[i][4] == "GYD": data[i][4]="BAK"
#     if data[i][2] == "OLH": data[i][2]="6R7"
#     if data[i][4] == "OLH": data[i][4]="6R7"
#     if data[i][2] == "NIB": data[i][2]="GGV"
#     if data[i][4] == "NIB": data[i][4]="GGV"
#     if data[i][2] == "MNT": data[i][2]="51Z"
#     if data[i][4] == "MNT": data[i][4]="51Z"
#     if data[i][2] == "IXC": data[i][2]="CHD"
#     if data[i][4] == "IXC": data[i][4]="CHD"
#     if data[i][2] == "LMA": data[i][2]="MTM"
#     if data[i][4] == "LMA": data[i][4]="MTM"
#     if data[i][2] == "MLH": data[i][2]="BSL"
#     if data[i][4] == "MLH": data[i][4]="BSL"
#     if data[i][2] == "SCE": data[i][2]="UNV"
#     if data[i][4] == "SCE": data[i][4]="UNV"
#     if data[i][2] == "SCE": data[i][2]="UNV"
#     if data[i][4] == "SCE": data[i][4]="UNV"
#     if data[i][2] == "LLB": continue
#     if data[i][4] == "LLB": continue
#     if data[i][2] == "\\N": continue
#     if data[i][4] == "\\N": continue
#     #if data[i][2] == str(47,'ascii', 'ignore'):  i = i+1
#     #if data[i][4] == "\N":  i = i+1
#     if data[i][2] == "UST": data[i][2]="SGJ"
#     if data[i][4] == "UST": data[i][4]="SGJ"
```

```
#    if data[i][2] == "IOQ": continue
#    if data[i][4] == "IOQ": continue
#    if data[i][2] == "KUZ": continue
#    if data[i][4] == "KUZ": continue
#    if data[i][2] == "TQI": continue
#    if data[i][4] == "TQI": continue
#    if data[i][2] == "AOQ": continue        #H
#    if data[i][4] == "AOQ": continue
#    if data[i][2] == "IUI": continue        #H
#    if data[i][4] == "IUI": continue
#    if data[i][2] == "QJH": continue         #H
#    if data[i][4] == "QJH": continue
#    if data[i][2] == "XEQ": continue         #H
#    if data[i][4] == "XEQ": continue
#    if data[i][2] == "JUK": continue         #H
#    if data[i][4] == "JUK": continue
#    if data[i][2] == "JUU": continue         #H
#    if data[i][4] == "JUU": continue
#    if data[i][2] == "KGQ": continue #H
#    if data[i][4] == "KGQ": continue
#    if data[i][2] == "KGH": continue #H
#    if data[i][4] == "KGH": continue
#    if data[i][2] == "KHQ": continue #H
#    if data[i][4] == "KHQ": continue
#    if data[i][2] == "NSQ": continue         #H
#    if data[i][4] == "NSQ": continue
#    if data[i][2] == "TQA": continue #H
#    if data[i][4] == "TQA": continue
#    if data[i][2] == "JAV": continue         #H
#    if data[i][4] == "JAV": continue
#    if data[i][2] == "SRK": continue
#    if data[i][4] == "SRK": continue
#    if data[i][2] == "NIQ": continue
#    if data[i][4] == "NIQ": continue
#    if data[i][2] == "SVR": continue
#    if data[i][4] == "SVR": continue
#    if data[i][2] == "KGT": continue
#    if data[i][4] == "KGT": continue
#    if data[i][2] == "ZGS": continue
#    if data[i][4] == "ZGS": continue
#    if data[i][2] == "ZBL" or data[i][4] == "ZBL": continue
#    if data[i][2] == "OSM" or data[i][4] == "OSM": continue
#    if data[i][2] == "THD" or data[i][4] == "THD": continue
#    if data[i][2] == "TGP" or data[i][4] == "TGP": continue
#    count = count +1
#    icao_orig = iata2icao[data[i][2]]
#    icao_dest = iata2icao[data[i][4]]
#    if icao_orig in icao2node_eu.keys():
#        if icao_dest in icao2node_eu.keys():
#            node_orig = icao2node_eu[icao_orig]
#            node_dest= icao2node_eu[icao_dest]
#            #if icao_orig == 'LEBL': print (icao2node_eu[icao_orig])
#            E.add_edge(node_orig,node_dest)
#    node_orig = ''
#    node_dest = ''
```

```
#     if icao_orig in icao2node_usa.keys():
#         if icao_dest in icao2node_usa.keys():
#             node_orig = icao2node_usa[icao_orig]
#             node_dest= icao2node_usa[icao_dest]
#             #if icao_orig == 'LEBL': print (icao2node_eu[icao_orig])
#             U.add_edge(node_orig,node_dest)

##########################################################################
        ###########
################################  RUTES
        ########################################
##########################################################################
        ###########

E_U_Di = nx.DiGraph()
icao2node_E_U_Di = {}
pos_E_U_Di = {}


j=1
path = 'files/logQuery/Mid2017.xlsx'
print ('Opening Opensky Network Routes -> ' + path + ': ' + str(os.path.exists(path)))

wall = xlrd.open_workbook(path)
sheet = wall.sheet_by_index(0)
data = [[sheet.cell_value(r,c) for c in range(sheet.ncols)] for r in range(sheet.nrows)]

icao_orig=''
icao_dest=''
count=0
for i in range(1,sheet.nrows):
    icao_orig = data[i][0]
    icao_dest = data[i][1]
    count = data[i][2]
    if icao_orig in icao2node_eu.keys():
        if icao_dest in icao2node_eu.keys():
            node_orig = icao2node_eu[icao_orig]
            node_dest = icao2node_eu[icao_dest]
            #if icao_orig == 'LEBL': print (icao2node_eu[icao_orig])
            E.add_edge(node_orig,node_dest,weight=count)



    if icao_orig in icao2node_usa.keys():
        if icao_dest in icao2node_usa.keys():
            node_orig = icao2node_usa[icao_orig]
            node_dest = icao2node_usa[icao_dest]
            U.add_edge(node_orig,node_dest,weight=count)

 # important retallat tornar a posar
    #if dict_airp_all_of_them[icao_orig].type != "heliport" and
        (dict_airp_all_of_them[icao_orig].continent == "EU" or
        dict_airp_all_of_them[icao_orig].country == "US"):
```

```
        #if dict_airp_all_of_them[icao_dest].type != "heliport" and
         (dict_airp_all_of_them[icao_dest].continent == "EU" or
         dict_airp_all_of_them[icao_dest].country == "US"):
            #if icao_orig not in icao2node_E_U_Di.keys():
                #icao2node_E_U_Di.update({icao_orig:j})
                #j=j+1
            #if icao_dest not in icao2node_E_U_Di.keys():
                #icao2node_E_U_Di.update({icao_dest:j})
                #j=j+1


    # a eliminar (prova)
    #if icao_orig in icao2node_eu or icao_orig in icao2node_usa:
        #if icao_dest in icao2node_eu or icao_dest in icao2node_usa:
            #if icao_orig not in icao2node_E_U_Di.keys():
                #icao2node_E_U_Di.update({icao_orig:j})
                #j=j+1
            #if icao_dest not in icao2node_E_U_Di.keys():
                #icao2node_E_U_Di.update({icao_dest:j})
                #j=j+1


#################################################################
#################################################################
print('#########################################################')
print ('Graph E (not connected graph)-> Nodes: ' + str(E.order()) + ', Edges: ' +
        str(len(E.edges())))
print ('Graph U (not connected graph)-> Nodes: ' + str(U.order()) + ', Edges: ' +
        str(len(U.edges())))
#print ('Graph E_U (not connected graph)-> Nodes: ' + str(E_U.order()) + ', Edges: ' +
        str(len(E_U.edges())))
print('#########################################################')

# Make E graphs connected.
for i in E.nodes():
    node_edgelist = E.edges([i])     #lista de las conexiones de cada node
    if not node_edgelist:           # si no tiene lista, se elimina el node
        E.remove_node(i)
try:
    E.remove_node(icao2node_eu['ENSD'])
    E.remove_node(icao2node_eu['ENFL'])
except:
    pass

for i in E.nodes():
    pos_E[i] = m(E.node[i]['lon'], E.node[i]['lat'])

# Make U graphs connected.
for i in U.nodes():
    node_edgelist = U.edges([i])     #lista de las conexiones de cada node
    if not node_edgelist:           # si no tiene lista, se elimina el node
        U.remove_node(i)

for i in U.nodes():
    pos_U[i] = m(U.node[i]['lon'], U.node[i]['lat'])
```

```
print(nx.is_connected(E))
print(nx.is_connected(U))


############## joint into global graph
GLOBAL = nx.Graph()
GLOBAL = nx.disjoint_union(E,U)
icao2node_GLOBAL = {}

for n in range(len(GLOBAL.nodes())):
    icao2node_GLOBAL[GLOBAL.node[n]['icao']] = n
    if GLOBAL.node[n]['icao'] == 'LFRN':
        a=1


#E_U_Di.add_nodes_from(icao2node_E_U_Di.values())
#for key in icao2node_E_U_Di:
#    i= icao2node_E_U_Di[key]
#    E_U_Di.add_node(i, icao = key, iata = dict_airp_all_of_them[key].iata, lon =
        float(dict_airp_all_of_them[key].lon), lat= float(dict_airp_all_of_them[key].lat),
        type = dict_airp_all_of_them[key].type, country =
        dict_airp_all_of_them[key].country, continent =
        dict_airp_all_of_them[key].continent, annualFlights = 0, annualFlightsRadar = 0)
        #else: pass
#    pos_E_U_Di[i] = m(E_U_Di.node[i]['lon'], E_U_Di.node[i]['lat'])


#for n in E_U_Di.node.items():
    #sumFlights = 0
    #for e in n.edge.items():
    #sumFlights = sumFlights + e.data


j=1
path = 'files/logQuery/Mid2017.xlsx'
#print (os.path.exists(path))

wall = xlrd.open_workbook(path)
sheet = wall.sheet_by_index(0)

data = [[sheet.cell_value(r,c) for c in range(sheet.ncols)] for r in range(sheet.nrows)]

icao_orig=''
icao_dest=''
count=0
for i in range(1,sheet.nrows):
    icao_orig = data[i][0]
    icao_dest = data[i][1]
    count = data[i][2]
    if (icao_orig in icao2node_eu.keys() and icao_dest in icao2node_usa.keys()) or
        (icao_orig in icao2node_usa.keys() and icao_dest in icao2node_eu.keys()):
        node_orig = icao2node_GLOBAL[icao_orig]
```

```
        node_dest = icao2node_GLOBAL[icao_dest]
        if GLOBAL.has_edge(node_orig,node_dest):
            w = GLOBAL.get_edge_data(node_orig,node_dest).get('weight')
            if count > w:
                GLOBAL.remove_edge(node_orig,node_dest)
                GLOBAL.add_edge(node_orig,node_dest,weight=count)
        else:
            GLOBAL.add_edge(node_orig,node_dest,weight=count)


for n in GLOBAL.nodes():
    n_edgeList= GLOBAL.edges(n)
    for e in n_edgeList:
        node_orig = e[0]
        node_dest = e[1]
        if GLOBAL.get_edge_data(node_orig,node_dest).get('weight') == None:
            GLOBAL.remove_edge(node_orig,node_dest)


# Remove self edges (loop)
for e in GLOBAL.edges():
    if e[0] == e[1]:
        GLOBAL.remove_edge(e[0],e[1])

# Actualitza el nombre total de vols.
#print(G.edges(1776,data=True))
c=0
for n in GLOBAL.nodes():
    totalsum=0
    for e in GLOBAL.edges(n):
        node_orig = e[0]
        node_dest=e[1]
        totalsum = totalsum +
         GLOBAL.get_edge_data(node_orig,node_dest).get('weight')

    if totalsum == 0:
        c=c+1
        print (c)
        GLOBAL.remove_node(n)
        continue
    # Aplica el ratio y crea variable flights2
    r = GLOBAL.node[n]['annualFlights']/totalsum
    total_f=0
    for e in GLOBAL.edges(node_orig,data=True):
        node_orig = e[0]
        node_dest=e[1]
        f = e[2].get('weight')
        GLOBAL.remove_edge(node_orig,node_dest)
        w = ceil(f*r)
        total_f = total_f + w
        GLOBAL.add_edge(node_orig,node_dest,weight=w)

    GLOBAL.node[n]['flights2'] = total_f
    GLOBAL.node[n]['cascadeFailure'] = 'False'
    GLOBAL.node[n]['checked'] = 'False'
```

```
    GLOBAL.node[n]['transfered'] = 0

    #if icao_orig in icao2node_E_U_Di.keys():
        #E_U_Di.node[icao2node_E_U_Di[icao_orig]]['annualFlightsRadar'] =
         E_U_Di.node[icao2node_E_U_Di[icao_orig]]['annualFlightsRadar'] + count
        #if icao_dest in icao2node_E_U_Di.keys():
            #node_orig = icao2node_E_U_Di[icao_orig]
            #node_dest = icao2node_E_U_Di[icao_dest]
            #if node_orig == 17 and node_dest==18:
                #abc=1
            #if node_orig == 18 and node_dest==17:
                #abc=1

            #E_U_Di.add_edge(node_orig,node_dest,weight=count)
             # #E_U_Di.node[node_orig]['annualFlightsRadar'] =
          E_U_Di.node[node_orig]['annualFlightsRadar'] + count
            #if icao_orig in icao2node_eu.keys():
                #E_U_Di.node[node_orig]['annualFlights'] =
          E.node[icao2node_eu[icao_orig]]['annualFlights']
            #if icao_dest in icao2node_eu.keys():
                #E_U_Di.node[node_dest]['annualFlights'] =
          E.node[icao2node_eu[icao_dest]]['annualFlights']
            #if icao_orig in icao2node_usa.keys():
                #E_U_Di.node[node_orig]['annualFlights'] =
          U.node[icao2node_usa[icao_orig]]['annualFlights']
            #if icao_dest in icao2node_usa.keys():
                #E_U_Di.node[node_dest]['annualFlights'] =
          U.node[icao2node_usa[icao_dest]]['annualFlights']

#E_U = nx.Graph()
#icao2node_E_U = icao2node_E_U_Di
#w1=0
#w2=0
#E_U.add_nodes_from(E_U_Di.nodes())
#for node in E_U.nodes():
    #E_U.node[node] = E_U_Di.node[node]

#for n1 in E_U_Di.nodes():
    #for n2 in E_U_Di.nodes():
        #try:
            #w1 = E_U_Di.get_edge_data(n1,n2).get('weight')
            #w2 = E_U_Di.get_edge_data(n2,n1).get('weight')
            #if w1>=w2:
                #E_U.add_edge(n1,n2,weight=w1,corrected='No')
            #else:
                #E_U.add_edge(n2,n1,weight=w2,corrected='No')
        #except:
            #pass



#correccio annualflights
#for n in E_U.nodes():
#    E_U.node[n]['annualFlights'] = E_U.node[n]['annualFlights'] -
         E_U.node[n]['annualFlights']*0.05
```

```
# ------------------------------------------------
#for n in E_U.nodes():
    #c=0
    #aa=''
    #list1=E_U.edges_iter(n,data='weight')
    #for aa in list1:
        #c=c+aa[2]
    #E_U.node[n]['flights'] = round(c/0.65*2)

#list_err1=[]
#list_err2=[]
#list_err3=[]
#for n in E_U.nodes():
    #a= E_U.node[n]['annualFlights']
    #b= E_U.node[n]['annualFlightsRadar']
    #if a==0 and b==0:
        #E_U.remove_node(n)
        #continue
    #if a>b:
        #err = ((a-E_U.node[n]['flights'])/a)*100
        #if err>10:
            #list_err1.append(n)
        #if err>20:
            #list_err2.append(n)
        #if err>30:
            #list_err3.append(n)
    #else:
        #err = ((b-E_U.node[n]['flights'])/b)*100
        #if err>10:
            #list_err1.append(n)
        #if err>20:
            #list_err2.append(n)
        #if err>30:
            #list_err3.append(n)
#print(len(list_err1))
#print(len(list_err2))
#print(len(list_err3))
# ----------------------------------------------------------------


#
# aF=0
# aFR=0
# ratio=0
# icao=''
# node_orig=''
# node_dest=''
# data=0
# data2=0
# list_E = list(E.edges_iter(data='weight'))
# list_U = list(U.edges_iter(data='weight'))
# listantes = list(E_U_Di.edges_iter(17,data='weight'))
# for n in E_U.nodes():
#     if n==17:
```

```
#         abc=1
#     if n==18:
#         abc=2
#     icao = E_U.node[n]['icao']
#     aF = E_U.node[n]['annualFlights']
#     aFR = E_U.node[n]['annualFlightsRadar']
#
#     c=0
#     list1 = E_U.edges(n,data='weight')
#     for f in list1:
#         c=c+f[2]
#
#     totalWF=c
#
#     if aF!=0 and aFR!=0:
#         if totalWF == 0:
#             totalWF=1
#         if aF>aFR:
#             ratio =round(aF/totalWF,10)
#         else:
#             ratio =round(aFR/totalWF,10)
#         if E_U.node[n]['icao'] == 'LEBL':
#             listlebl1 = E_U.edges(n,data='weight')
#
#         #E_U_Di.node[n]['annualFlightsRadar']  =
        round(E_U_Di.node[n]['annualFlightsRadar'] * ratio)
#         list2 = E_U.edges(n,data='weight')
#         for f in list2:
#             node_orig = f[0]
#             node_dest = f[1]
#             if node_orig == 17 and node_dest==18:
#                 abc=1
#             if node_orig == 18 and node_dest==17:
#                 abc=12
#
#             data = f[2]
#             data2 = round(data*ratio)
#             #print ("AF: " + str(aF) + ", AFR: " + str(aFR) + ", ratio: " +str(ratio))
#             #if E_U.node[n]['icao'] == 'LEBL':
#                 #print ("antes: " + str(E_U_Di.get_edge_data(node_orig,node_dest)) + "
        ratio: " + str(ratio))
#             E_U.remove_edge(node_orig,node_dest)
#             E_U.add_edge(node_orig,node_dest,weight=data2)
#             #if E_U.node[n]['icao'] == 'LEBL':
#                 #listlebl2 = E_U_Di.edges(n,data='weight')
#                 #print ("despues: " + str(E_U_Di.get_edge_data(node_orig,node_dest)))
# listadespues = list(E_U.edges_iter(17,data='weight'))


#### only 1 run

# for e in range(1,len(E_U.edges())+1):
#     #print (e)
#     for i in range(1, len(E_U.edges())+1):
#         #print(i)
```

```
#         try:
#             E_U.edge[e][i]['corrected'] = 'No'
#         except:
#             pass

# ----------------------------------------------------------------
#aF=0
#aFR=0
#ratio=0
#icao=''
#node_orig=''
#node_dest=''
#data=0
#data2=0
#list_E = list(E.edges_iter(data='weight'))
#list_U = list(U.edges_iter(data='weight'))
#listantes = list(E_U_Di.edges_iter(17,data='weight'))
#for n in E_U.nodes():
    #if n==17:
        #abc=1
    #if n==18:
        #abc=2
    #icao = E_U.node[n]['icao']
    #aF = E_U.node[n]['annualFlights']
    #aFR = E_U.node[n]['annualFlightsRadar']

    #c=0
    #list1 = E_U.edges(n,data='weight')
    #for f in list1:
        #c=c+f[2]

    #totalWF=c

    #if aF!=0 and aFR!=0:
        #if totalWF == 0:
            #totalWF=1
        #if aF>aFR:
            #ratio =round(aF/totalWF,10)
        #else:
            #ratio =round(aFR/totalWF,10)
        #if E_U.node[n]['icao'] == 'LEBL':
            #listlebl1 = E_U.edges(n,data='weight')

        # #E_U_Di.node[n]['annualFlightsRadar']  =
         round(E_U_Di.node[n]['annualFlightsRadar'] * ratio)
        #list2 = E_U.edges(n,data='weight')
        #for f in list2:
            #node_orig = f[0]
            #node_dest = f[1]
            #if node_orig == 17 and node_dest==18:
                #abc=1
            #if node_orig == 18 and node_dest==17:
                #abc=12

            #data = f[2]
```

```
            #data2 = round(data*ratio)
            # #print ("AF: " + str(aF) + ", AFR: " + str(aFR) + ", ratio: " +str(ratio))
            # #if E_U.node[n]['icao'] == 'LEBL':
                # #print ("antes: " + str(E_U_Di.get_edge_data(node_orig,node_dest)) + "
          ratio: " + str(ratio))
            #if E_U[node_orig][node_dest]['corrected']== 'No':
                #E_U.remove_edge(node_orig,node_dest)
                #E_U.add_edge(node_orig,node_dest,weight=data2,corrected='Yes')
                # #E_U[node_orig][node_dest]['corrected'] = 'Yes'
                # #if E_U.node[n]['icao'] == 'LEBL':
                    # #listlebl2 = E_U_Di.edges(n,data='weight')
                    # #print ("despues: " + str(E_U_Di.get_edge_data(node_orig,node_dest)))


#listadespues = list(E_U.edges_iter(17,data='weight'))

# ----------------------------------------------------------------




print('############################################################')
print ('Graph E (connected graph)-> Nodes: ' + str(E.order()) + ', Edges: ' +
        str(len(E.edges())))
print ('Graph U (connected graph)-> Nodes: ' + str(U.order()) + ', Edges: ' +
        str(len(U.edges())))
print ('Graph GLOBAL (connected graph)-> Nodes: ' + str(GLOBAL.order()) + ', Edges:
        ' + str(len(GLOBAL.edges())))
print('############################################################')
print ('')
print('Graph E info: ' + nx.info(E))
print('Graph U info: ' + nx.info(U))
print('Graph GLOBAL info: ' + nx.info(GLOBAL))

path = 'out/'
nx.write_gpickle(E, path + 'E.pkl')
nx.write_gpickle(U, path + 'U.pkl')
nx.write_gpickle(GLOBAL, path + 'GLOBAL.pkl')
nx.write_gml(GLOBAL, path + 'GLOBAL.gml')
nx.write_gml(U, path + 'U.gml')
nx.write_gml(E, path + 'E.gml')
print('E.pkl, U.pkl, GLOBAL.pkl generated successful.')


#print (E_U.node[17])    #LEBL
#print (E_U.node[16])    #LEMD
#print (E_U.node[4])     #KLAX
#print (E_U.node[5])     #KJFK
#print (E_U.node[6])     #KSFO
#print (E_U.node[icao2node_E_U['KLGA']])     #KLGA
#print (E_U.node[icao2node_E_U['KATL']])     #KATL
#print (E_U.node[icao2node_E_U['KSEA']])     #KSEA
#print (E_U.node[icao2node_E_U['KBOS']])     #KBOS
```

```
# sys.stdout = old_stdout
# log_file.close()



#
#
#
# ################### Draw Graph EUROPE ##################
# E_edgeList = [(u, v) for (u, v, d) in E.edges(data=True)]
#
# nx.draw_networkx_nodes(E, pos_E, node_size = 10, node_shape='.', node_color='b',
        label='Europe Airports')
# nx.draw_networkx_edges(E, pos_E, node_size = 10, node_shape='.', node_color='b',
        width=0.05)
# #nx.draw_networkx_edges(E, pos_E,edgelist=E_edgeList)
#
#
# #nx.draw_networkx_labels(E,pos_E, font_size=7, font_color='green')
#
# # labels = [left,right,top,bottom]
# m.drawparallels(np.arange(-80,81,20.),labels=[False,True,False,False],
        linewidth=0.0, fontsize=8)
# meridians = m.drawmeridians(np.arange(-
        181.,179.,20.),labels=[True,False,False,True], linewidth=0.0, fontsize=8)
# for m in meridians:                #Rotació del label dels meridians 45graus
#     try:
#         meridians[m][1][0].set_rotation(45)
#     except:
#         pass
#
# plt.title('Europe Airports')
# plt.legend(numpoints = 3, loc=4)
# # rect = [left, bottom, right, top]
# fig1.tight_layout(rect=[0,0.04,1,1])
# plt.rcParams["figure.figsize"] = [7,5]
#
# plt.show(fig1)
# plt.savefig(path_out +"E-basemap-airports.png")
#
# #print("--- %s seconds to Graph NX EUROPE ---" % (time.time() - start_time))
#
#
#
# ################### Draw Graph NA i usa ##################
# fig2 = plt.figure(2)
#
# m2.drawcountries(linewidth = 0.5)
# #m.fillcontinents(color='white',lake_color='white')
# m2.fillcontinents()
# m2.drawcoastlines(linewidth=0.5)
#
# nx.draw_networkx_nodes(U, pos_U, node_size = 10, node_shape='.', node_color='b',
        label='USA Airports')
```

```
# nx.draw_networkx_edges(U, pos_U, node_size = 10, node_shape='.', node_color='b',
        width=0.025)
#
#
#
# #nx.draw_networkx_nodes(U, pos_U, node_size = 10, node_shape='.',
        node_color='r', label='USA Airports')
# #nx.draw_networkx_nodes(NA, pos_NA, node_size = 10, node_shape='.',
        node_color='r', label='NA Airports')
# #nx.draw_networkx_labels(U,pos_U, font_size=8, font_color='green')
#
# # labels = [left,right,top,bottom]
# m2.drawparallels(np.arange(-80,81,20.),labels=[False,True,False,False],
        linewidth=0.0, fontsize=8)
# meridians = m2.drawmeridians(np.arange(-
        181.,179.,20.),labels=[True,False,False,True], linewidth=0.0, fontsize=8)
# for m2 in meridians:                  #Rotació del label dels meridians 45graus
#     try:
#         meridians[m2][1][0].set_rotation(45)
#     except:
#         pass
#
# plt.title('USA Airports')
# plt.legend(numpoints = 3, loc=4)
# # rect = [left, bottom, right, top]
# fig2.tight_layout(rect=[0,0.04,1,1])
# plt.rcParams["figure.figsize"] = [7,5]
#
# plt.show(fig2)
# plt.savefig(path_out +"U-basemap-airports.png")
#
# ################### Draw Graph E_U_Di ##################
# fig3 = plt.figure(3)
#
# m3.drawcountries(linewidth = 0.5)
# #m.fillcontinents(color='white',lake_color='white')
# m3.fillcontinents()
# m3.drawcoastlines(linewidth=0.5)
#
# nx.draw_networkx_nodes(E_U_Di, pos_E_U_Di, node_size = 10, node_shape='.',
        node_color='b', label='E & U Airports')
# nx.draw_networkx_edges(E_U_Di, pos_E_U_Di, node_size = 10, node_shape='.',
        node_color='b', width=0.025)
#
#
#
# #nx.draw_networkx_nodes(U, pos_U, node_size = 10, node_shape='.',
        node_color='r', label='USA Airports')
# #nx.draw_networkx_nodes(NA, pos_NA, node_size = 10, node_shape='.',
        node_color='r', label='NA Airports')
# #nx.draw_networkx_labels(U,pos_U, font_size=8, font_color='green')
#
# # labels = [left,right,top,bottom]
# m3.drawparallels(np.arange(-80,81,20.),labels=[False,True,False,False],
        linewidth=0.0, fontsize=8)
```

```
# meridians = m3.drawmeridians(np.arange(-
        181.,179.,20.),labels=[True,False,False,True], linewidth=0.0, fontsize=8)
# for m3 in meridians:                #Rotació del label dels meridians 45graus
#     try:
#         meridians[m3][1][0].set_rotation(45)
#     except:
#         pass
#
# plt.title('E & U Airports')
# plt.legend(numpoints = 3, loc=4)
# # rect = [left, bottom, right, top]
# fig3.tight_layout(rect=[0,0.04,1,1])
# plt.rcParams["figure.figsize"] = [7,5]
#
# plt.show(fig3)
# plt.savefig(path_out +"E_U_Di-basemap-airports.png")
#
#
#
#
#
#
#
# #print("--- %s seconds to Graph NX USA ---" % (time.time() - start_time))
#
#
# # fig3 = plt.figure(3)
# # nx.draw_networkx(E, pos_E, node_size = 10, node_shape='.', node_color='b',
        label='Europe Airports')
# # nx.draw_networkx_edges(E,pos_E,edgelist=E_edgeList,width=0.05)
# # #nx.draw_networkx_edges(E, pos_E, node_size = 10, node_shape='.',
        node_color='b', width=0.05)
# # plt.show(fig3)
#
# fig4 = plt.figure(4)
# nx.draw_networkx_nodes(E, pos_E, node_size = 10, node_shape='.', node_color='b',
        label='Europe Airports')
# nx.draw_networkx_edges(E, pos_E, node_size = 10, node_shape='.', node_color='b',
        width=0.025)
# plt.show(fig4)
# plt.savefig(path_out +"E-airports.png")
#
# fig5 = plt.figure(5)
# nx.draw_networkx_nodes(U, pos_U, node_size = 10, node_shape='.', node_color='b',
        label='USA Airports')
# nx.draw_networkx_edges(U, pos_U, node_size = 10, node_shape='.', node_color='b',
        width=0.025)
# plt.show(fig5)
# plt.savefig(path_out +"U-airports.png")
#
# fig6 = plt.figure(6)
# nx.draw_networkx_nodes(E_U_Di, pos_E_U_Di, node_size = 10, node_shape='.',
        node_color='b', label='USA Airports')
# nx.draw_networkx_edges(E_U_Di, pos_E_U_Di, node_size = 10, node_shape='.',
        node_color='b', width=0.025)
```

```
# plt.show(fig6)
# plt.savefig(path_out +"E_U_Di-airports.png")
#
# ########################################
# ##################### pintar aeroports no utils vs aeroports utils, weighted graph
#
#
#
# ################################################################
# # E.node[67] -> info LEBL node
# # E.node[67]['annualFlights']
# # E.neighbors(67)
# # E.edges(67)
# # E.edges([67,157])
# # E.get_edge_data(67,157)
# ################################################################
```